Les Grands Progiciels

ARCHIVE

Progiciel de Bases de Données

E. TENIN

sur SINCLAIR QL

sur IBM PC et compatibles

Edimicro

ARCHIVE ™

PROGICIEL DE BASES DE DONNÉES

ARCHIVE ™

PROGICIEL DE BASES DE DONNÉES

par

Eric TENIN

Collection « les grands progiciels »

Edimicro

Dans la même série, aux Editions Edimicro :

- Guide pratique du Sinclair QL, par Eric Tenin
- Abacus, Easel et Quill sur Sinclair QL, par Jean-Marie Devisscher
- Intelligence Artificielle sur Sinclair QL, par Keith et Steven Brain

Tous les noms de marque qui sont cités dans cet ouvrage sont des marques déposées ™

© 1985 F.D.S./Edimicro. Première édition.
Couverture : Thierry Duval

Imprimé en France. Droits mondiaux réservés.

« La loi du 11 mars 1957 n'autorisant, aux termes des alinéas 2 et 3 de l'article 41, d'une part, que les "copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à une utilisation collective", et d'autre part, que les analyses et les courtes citations dans un but d'exemple et d'illustration, "toute représentation ou reproduction intégrale, ou partielle, faite sans le consentement de l'auteur ou de ses ayants droit ou ayants cause, est illicite" (alinéa 1er de l'article 40). » « Cette représentation ou reproduction, par quelque procédé que ce soit, constituerait une contrefaçon sanctionnée par les articles 425 et suivants du Code pénal. »

© 1985 F.D.S./Edimicro
ISBN 2-904457-39-9

EDIMICRO

DÉPARTEMENT ÉDITIONS DE F.D.S. SARL

121/127, avenue d'Italie, 75013 Paris

Table des matières

Nos de pages

CHAPITRE 1 : **ARCHIVE** 13

1.1. Généralités sur les bases de données 13
1.2. Archive 14
1.3. Les applications 15
1.4. Fichiers papier et fichiers informatisés 15

1.4.1. Les avantages 15
1.4.2. Les contraintes 16

1.5. Installation 18

1.5.1. Sur le QL 18
1.5.2. Sur l'IBM PC 19

CHAPITRE 2 : **PRISE EN MAIN** 25

2.1. Présentation 25
2.2. L'éditeur 29
2.3. Prise de contact 29

2.3.1. Les commandes à connaître absolument 29
2.3.2. Création d'un fichier 30
2.3.3. Insertion, modification, suppression 31
2.3.4. Opération de recherches 33
2.3.5. Ouverture et fermeture d'un fichier 36
2.3.6. Sélection et tri 38
2.3.7. Communication entre les logiciels et copies 39

Nos de pages

2.3.8. Les autres commandes 42
2.3.9. Les accès à la mémoire de masse 45

2.4. Création pratique d'un fichier 49

2.4.1. Les champs 49
2.4.2. Remplissage du fichier 50

2.5. Utilisation de quelques commandes simples 51

2.5.1. Recherche d'une fiche 54
2.5.2. Sélection d'un paquet de fiches 57
2.5.3. Tri, ordonnancement 57
2.5.4. Modifications manuelles 59
2.5.5. Modifications automatiques 60

2.6. Utilisation de quelques commandes plus complexes 61

2.6.1. Introduction aux fonctions 61
2.6.2. Les opérations 64
2.6.3. Les autres fonctions 67

2.7. Les masques d'écran 88

2.7.1. Création d'un masque d'écran 88
2.7.2. Les commandes associées aux masques d'écran 92
2.7.3. Application pratique 93

CHAPITRE 3 : **PROGRAMMATION** 97

3.1. Pourquoi la nécessité de programmer ? 97
3.2. Les instructions du langage de programmation 98
3.3. Comment programmer ARCHIVE 106
3.4. Des procédures plus sophistiquées 109

3.4.1. Simplifier les commandes 109
3.4.2. Automatisation des tâches 115

3.5. Remarques supplémentaires sur les procédures 118

3.5.1. Traitement des erreurs 118
3.5.2. La commande exéc 121
3.5.3. Les paramètres 121
3.5.4. Les variables locales 122
3.5.5. La récursivité 123

3.6. Utilisation multifichiers 123

CHAPITRE 4 : **EXEMPLES D'APPLICATIONS** 127

4.1. Qui se sert d'ARCHIVE, quand, pendant combien de temps ? 128

4.1.1. Le fichier « mouchard » 128
4.1.2. Un accès limité 129

4.2. Une application complète pour un problème quotidien 134

Nos de pages

CHAPITRE 5 : **UNE APPLICATION ORIGINALE** 145

5.1. Questionnaire et dépouillement sur ARCHIVE 145

5.1.1. Les données du problème 146
5.1.2. La solution informatique 146

5.2. Le questionnaire 146

5.2.1. La solution informatique 146

5.3. Le dépouillement 155

ANNEXES 161

A : Programme "Boot" 161
B : Un programme de publi-postage pour le QL 163
C : Quelques définitions de vocabulaire 171
D : Les différents modes d'archive 175

PRÉSENTATION

Nous avons conçu ce guide de façon très progressive, et — nous l'espérons — attractive.

Même si vous n'avez jamais touché un ordinateur de votre vie, vous pourrez exploiter aisément les innombrables possibilités d'ARCHIVE.

Les programmeurs avertis trouveront en outre des exemples d'application originaux, qui leur serviront à réaliser leurs propres programmes : comptabilité, dépouillement de questionnaire, publi-postage... Encouragés par l'accueil très favorable qu'a reçu notre « Guide pratique du Sinclair QL », aux mêmes éditions, nous souhaitons apporter le même service aux utilisateurs d'ARCHIVE, sur Sinclair QL ou sur IBM PC.

CHAPITRE 1

Archive

1.1. GÉNÉRALITÉS SUR LES BASES DE DONNÉES

On définit généralement le terme « bases de données » par un ensemble d'informations organisées de façon que l'utilisateur puisse facilement les retrouver, les classer, les trier, etc. Le fait de pouvoir effectuer toutes ces opérations conduit, à proprement parler, à « gérer » sa base de données, d'où le nom communément employé de « système de gestion de bases de données » (SGBD).

Une base de données n'est ni plus ni moins que la reproduction d'un fichier classique (constitué de fiches cartonnées) dans lequel sont consignées un certain nombre d'informations comme par exemple les noms, prénoms, adresses et numéros de téléphone. Les fichiers cartonnés sont généralement classés par ordre alphabétique et permettent de retrouver facilement une fiche. Chaque rubrique (nom, prénom, adresse, etc.) porte le nom de *champ* et chaque fiche est qualifiée *d'enregistrement*.

On distingue généralement trois types de SGBD :

— Hiérarchique

Système dans lequel chaque enregistrement comporte un champ *principal* et des champs *subordonnés* : il faut d'abord passer par le

champ principal pour accéder aux informations des champs subordonnés.

— Réseaux

Avec ce système il est possible de définir des liaisons entre des fichiers et il est par conséquent possible de retrouver des informations sans passer par les champs intermédiaires.

— Relationnel

Ce système est beaucoup plus souple d'emploi étant donné qu'il n'est besoin ni de passer par des champs intermédiaires (SGBD hiérarchiques), ni de définir des liaisons entre les fichiers (SGBD en réseaux). Tous les champs sont en effet facilement accessibles et peuvent servir à retrouver des informations.

1.2. ARCHIVE

ARCHIVE fait partie du dernier type de bases de données décrit. Non seulement il va être possible de retrouver des informations à partir de n'importe quel champ, mais cela sera rendu possible grâce à un langage très simple et très facile d'accès. En outre ARCHIVE dispose d'un *langage de programmation* qui permet à l'utilisateur de programmer certaines applications qui pourront alors être utilisées ultérieurement par n'importe quel néophyte ne connaissant pas nécessairement les commandes pour communiquer avec ARCHIVE.

ARCHIVE permet aussi d'ouvrir plusieurs fichiers en même temps (ex. : un fichier clients et fournisseurs) et de les utiliser simultanément.

Sur l'IBM PC, une option supplémentaire permet de se servir conjointement de QUILL (traitement de texte) et d'ARCHIVE pour les envois en publi-postage (*mailing*). Bien que cette option ne soit pas disponible sur le QL, nous avons inclus dans ce guide un petit programme qui permet d'effectuer ceci avec un minimum de contraintes.

ARCHIVE se distingue des autres systèmes de gestion de bases de données en ce sens qu'il est extrêmement simple d'utilisation et que quelques commandes suffisent pour le mettre en œuvre. De plus toutes les commandes sont en français et se rapprochent du langage naturel.

1.3. LES APPLICATIONS

A priori, ARCHIVE ne peut servir qu'à l'exploitation de fichiers somme toute assez classiques (fichiers clients, fournisseurs, répertoires, etc.) mais il permet en outre des applications beaucoup plus pointues comme par exemple la gestion d'un stock, la comptabilité, ou le dépouillement rapide d'un questionnaire ; toutes ces applications utilisent le même principe (enregistrement structuré) et permettent donc une utilisation non restrictive d'ARCHIVE. Ces applications seront décrites à la fin de ce guide.

1.4. FICHIERS-PAPIER ET FICHIERS INFORMATISÉS

En principe, les fichiers que vous avez déjà réalisés sur des fiches en papier cartonné sont relativement faciles à manipuler, ils ne sont pas fragiles et peuvent être consultés par le premier venu.

Un fichier informatisé requiert quelques précautions de manipulation ainsi qu'une légère connaissance de l'outil informatique, mais permet une souplesse sans pareil.

1.4.1. Les avantages

* *économie de temps*

Les fichiers informatisés permettent de gagner un temps précieux en manipulation, en rangement et en recherche. La seule tâche qui peut prendre plus de temps qu'un fichier manuel, c'est de saisir les informations au clavier. Une fois ceci effectué, il devient extrêmement simple de rechercher un nom ou bien une adresse, d'éditer des étiquettes selon un format quelconque ou de sélectionner certaines fiches selon des critères divers.

* *traitement de l'information*

Pour les envois en nombre, les postes françaises imposent que les lettres soient classées par département ; avec un jeu d'étiquettes habituel on ne peut se permettre d'avoir à la fois un classement par ordre alphabétique et par département. Avec ARCHIVE, en revanche, il devient très facile d'effectuer ce genre d'opération en un temps minimal. Il est aussi possible de rechercher n'importe quelle fiche très rapidement, on peut aussi

s'en servir pour analyser des données financières ou bien pour toutes sortes d'applications.

D'une manière générale ARCHIVE permet de réaliser toutes les applications bien plus facilement que sur un fichier normal ; il suffit pour cela de connaître quelques commandes.

1.4.2. Les contraintes

* *précautions*

Toutes les données seront stockées sur des micro-cassettes ou sur des disquettes ou encore sur un disque dur (pour l'IBM PC). Il est indispensable de manier ces supports magnétiques avec soin et en particulier de ne jamais les exposer à de trop fortes chaleurs ou à un champ magnétique.

* *copie*

Il est toujours préférable de faire systématiquement une copie de sécurité des fichiers afin d'avoir en permanence un double des données rentrées et éviter ainsi de perdre accidentellement le travail de saisie nécessaire à la constitution du fichier. Après chaque nouvelle opération, faites une copie complète de la nouvelle version du fichier. Nous expliquerons ultérieurement comment procéder.

* *structure d'un fichier informatisé*

La structure d'un fichier informatisé impose quelques contraintes qui n'existent pas pour les fichiers manuels :

— Il est indispensable de déterminer, avant de commencer, les différents champs qui composent le fichier.

Exemple :
nom
adresse
téléphone
etc.

Ces champs répertoriés, il ne sera pas facile d'en ajouter un, comme par exemple « prénom ». Il est par conséquent important de bien définir ses besoins, avant de créer un fichier.

— l'utilisateur doit définir, au moment de la création du fichier, si un champ est destiné à recevoir du texte ou bien des chiffres.

Exemple :

nom (texte)
téléphone (chiffres)
adresse (texte)
code postal (chiffres)

Afin de simplifier cette opération, il existe un signe distinctif qui permet de définir si un champ est destiné à recevoir du texte ou des chiffres. Ce signe est familier à ceux qui connaissent déjà le Basic ; tout nom de champ destiné à recevoir du texte doit être suivi du signe "$" (Dollar), aucun nom de champ destiné à recevoir des chiffres n'est suivi d'un signe distinctif.

Exemple :

nom$
téléphone
adresse$
code postal

Cela semble d'abord une contrainte importante, mais il s'agit simplement d'une convention, utile pour le traitement ultérieur de l'information.

* *commandes spécifiques à connaître absolument*

A l'inverse des fichiers écrits sur des cartes, où aucune connaissance particulière n'est nécessaire (si ce n'est de savoir lire !!!), la manipulation d'ARCHIVE impose que l'on sache utiliser certaines commandes spécifiques qui vont permettre de créer des fichiers, de les trier, de les modifier, etc. Ce sont des commandes simples que cet ouvrage s'efforcera d'exposer de façon claire et attractive.

LES SPÉCIFICATIONS D'ARCHIVE

Les champs : ils peuvent avoir treize caractères au maximum et ne doivent pas commencer par un chiffre (ils peuvent cependant en contenir). Ils doivent être composés des lettres a à z (en majuscules ou en minuscules) et ne doivent pas contenir de caractères diacritiques à l'exception de :
"." qui sert lors de l'ouverture simultanée de plusieurs fichiers
"_" qui sert à lier deux mots entre eux ex. No_compte
"$" qui sert à indiquer que le champ est alphanumérique (texte).

N.B. : les noms de variables obéissent aux mêmes règles. Les champs et les variables peuvent contenir au maximum 255 caractères.

Les enregistrements : ils sont composés d'un ou plusieurs champs numériques ou alphanumériques. Il peut y avoir un maximum de 255 champs par enregistrement et leur longueur n'a pas besoin d'être précisée.

Les fichiers : Un fichier est constitué d'un ou plusieurs enregistrements et porte un nom *physique* (exemple : clients) et un nom *logique* (exemple : "c") qui permettent l'utilisation multifichiers. Il peut y avoir en théorie un maximum de 65 000 enregistrements par fichier ; en fait ARCHIVE utilise la mémoire de masse (micro-cassettes, disquettes, disque dur) et c'est la capacité de ces derniers qui limite dans la pratique le nombre d'enregistrements.

NE RETIREZ JAMAIS LA MÉMOIRE DE MASSE (DISQUETTES OU MICRO-CASSETTES) AVANT D'AVOIR REFERMÉ VOS FICHIERS. DE MÊME NE DÉBRANCHEZ JAMAIS VOTRE APPAREIL AVANT D'AVOIR COMPLÈTEMENT TERMINÉ UNE SESSION.

1.5. INSTALLATION

Que cela soit sur l'IBM PC ou bien sur le QL, les logiciels doivent être copiés avant emploi. Ceci afin d'éviter de détériorer par erreur les originaux et de les rendre inutilisables par la suite.

1.5.1. Installation sur le QL

Après avoir mis l'ordinateur sous tension et appuyé sur F1 ou F2, mettez la micro-cassette contenant ARCHIVE dans le lecteur 2 (celui de droite) et une micro-cassette vierge dans le lecteur de gauche. Tapez ensuite la commande suivante :

```
lrun mdv2_clone
```

Aussitôt le lecteur de droite se met en marche et après quelques secondes un message vous invitant à taper sur la barre d'espace apparaît à l'écran. Ceci fait, le lecteur de gauche se met à son tour en marche et la cassette vierge se formate (préparation pour recevoir la copie d'ARCHIVE).

Cette opération dure quelques secondes, puis la copie proprement dite commence. Celle-ci dure environ trois minutes durant lesquelles les différents noms des fichiers qui sont copiés apparaissent à l'écran.

Ne retirez aucune cassette avant que les lecteurs ne se soient complètement arrêtés.

Lorsque l'opération est terminée, retirez l'original, vous n'en aurez plus besoin à moins que la copie ne soit endommagée d'une façon quelconque. La copie est maintenant prête à être utilisée.

RAPIDEMENT...

Mettez la cassette contenant ARCHIVE dans le lecteur de droite.
Mettez une cassette vierge dans le lecteur de gauche.
Tapez LRUN Mdv2_clone.
Appuyez sur la barre d'espace.
Attendez la fin de la copie.

1.5.2. Installation sur l'IBM PC

Sur l'IBM PC, le logiciel n'étant pas livré avec la machine, celui-ci est protégé contre les copies pirates. Il est en effet toujours nécessaire de garder la disquette originale sous la main *(disquette source)*, lorsque l'on veut utiliser ARCHIVE. L'utilisateur doit cependant faire une copie mais lors du chargement il faudra toujours introduire la disquette *source* au moment où le logiciel en fera la demande, afin que ce dernier vérifie qu'il ne s'agit pas d'une copie pirate.

Mettez l'ordinateur sous tension, en ayant soin d'introduire la disquette contenant le système d'exploitation (DOS) dans le lecteur A. Une fois ceci fait le prompt vous invite à rentrer une commande le prompt devrait normalement indiquer A>, si ce n'est pas le cas tapez :

a : <RETURN>

Retirez la disquette contenant le DOS du lecteur A, remplacez-la par la disquette portant la mention « utilitaires » et tapez :

installe <RETURN>

Une fois lues les informations apparaissant à l'écran, tapez sur une touche pour continuer.

XCHANGE vous demande alors de sélectionner l'unité sur laquelle vont se copier les fichiers destinés à constituer la disquette-programme ; à ce stade deux possibilités sont offertes :

— vous ne disposez que de deux unités de disquettes (A et B) et dans ce cas vous ne devez rien indiquer de particulier,

— vous disposez de plusieurs unités (et notamment d'un disque dur C) et vous devez préciser sur quelle unité vous souhaitez créer

la disquette-programme ; pour cela sélectionnez à l'aide des flèches se trouvant sur le pavé numérique l'option UNITÉ DE DESTINATION et appuyez sur <RETURN>. Au bas de l'écran apparaissent maintenant les unités disponibles pour effectuer l'opération ; il suffit d'en sélectionner une (toujours à l'aide des flèches du pavé numérique) et de taper sur <RETURN> pour confirmer votre choix.

En second lieu, il convient d'indiquer à XCHANGE quel type de moniteur vous utilisez (couleur ou monochrome) ; encore une fois deux choix se présentent :

— vous disposez d'un moniteur couleur, et dans ce cas vous n'avez rien à changer,

— vous disposez d'un moniteur monochrome, et vous devez alors le préciser ; appuyez sur <RETURN> et sélectionnez à l'aide des flèches l'option correspondant à votre cas, puis validez en tapant à nouveau sur <RETURN> .

Maintenant tout est prêt pour installer la disquette. Appuyez sur F3 pour voir apparaître les commandes du menu, puis sur « I » pour sélectionner l'option « installe ». Remplacez la disquette « utilitaire » se trouvant dans l'unité A par la disquette repérée « source », placez une disquette formatée dans le lecteur (B si vous installez le programme sur une disquette et C, si vous l'installez sur le disque dur) et tapez sur une touche. Plusieurs options apparaissent alors ; sélectionnez celle qui indique « création d'un nouveau système XCHANGE », et l'installation commence alors (les fichiers sont copiés d'une disquette vers l'autre).

Ceci fait, le système vous demande si vous voulez installer d'autres disquettes ; ceci est particulièrement utile lorsque vous avez acheté plusieurs modules séparément (par exemple : ARCHIVE et QUILL) ; si vous avez acheté les quatre logiciels, il n'y a qu'une seule disquette à installer et vous venez de le faire. Répondez N (pour non) si vous ne souhaitez pas installer d'autres disquettes.

En second lieu, XCHANGE vous invite à installer la disquette utilitaire (celle qui contient les fichiers d'aide que vous pourrez appeler à tout moment pendant l'exécution de vos tâches) ; tapez O, placez la disquette étiquetée « utilitaire » dans le lecteur A et tapez sur une touche.

A présent l'installation de la disquette « utilitaire » commence. Si vous avez effectué l'opération précédente sur une disquette (celle qui se trouve encore dans le lecteur B), il n'y aura pas assez de place pour copier les nouveaux fichiers, et par conséquent, il faudra insérer une nouvelle disquette formatée à sa place (si vous avez utilisé votre disque dur, et qu'il y a suffisamment de place dessus, il n'y a pas de manipulation à effectuer).

Tapez alors « Q » pour quitter l'utilitaire et revenir au système (insérez la disquette contenant le DOS dans le lecteur A).

L'installation proprement dite est maintenant terminée et ARCHIVE est prêt à l'emploi.

RAPIDEMENT...

— Mettez l'ordinateur sous tension et insérez le DOS dans le lecteur A si votre système ne démarre pas automatiquement sur le disque dur,

— tapez A : <RETURN> si le prompt n'indique pas A>

— introduisez la disquette « utilitaire » dans l'unité A et tapez :

installe <RETURN>

— tapez sur une touche

— sélectionnez l'unité sur laquelle doit se faire la copie (si vous disposez de plus de deux unités)

— sélectionnez le type de moniteur dont vous disposez (couleurs ou monochrome)

— insérez une disquette formatée dans l'unité de destination (si vous ne l'installez pas sur le disque dur)

— appuyez sur F3, tapez sur « I »

— remplacez la disquette « utilitaire » par la disquette « source »

— sélectionnez l'option « création d'un nouveau système » XCHANGE et tapez sur <RETURN> pour exécuter l'opération d'installation

— répétez l'opération avec d'autres disquettes-source si vous avez acheté les modules séparément (O pour oui/N pour non)

— si vous souhaitez installer une disquette « utilitaire » répondez O et remplacez la disquette du lecteur B par une nouvelle disquette formatée

— remplacez la disquette « source » par la disquette « utilitaire » et tapez sur une touche

— tapez sur « Q » pour revenir au système (insérez la disquette contenant le DOS dans le lecteur A

L'installation est terminée.

SUR LE QL

Pour utiliser ARCHIVE deux commandes sont possibles.

* Si les touches F1 ou F2 ont déjà été enfoncées (juste après la mise sous tension de l'appareil), tapez la commande suivante :

```
lrun mdv1_boot
```

LA CASSETTE CONTENANT ARCHIVE DOIT TOUJOURS ÊTRE PLACÉE DANS LE LECTEUR DE GAUCHE (mdv1-)

le lecteur se met alors en marche et ARCHIVE est chargé en machine

* Si les touches F1 ou F2 n'ont pas encore été enfoncées (c'est-à-dire juste après la mise sous tension ou après un "reset"), tapez simplement sur l'une ou l'autre de ces deux touches, selon que vous disposez d'un moniteur ou d'une télévision, et ARCHIVE est automatiquement chargé en machine.

Dans les deux cas, après quelques secondes apparaît à l'écran le message suivant :

ARCHIVE
base de données
Copyright Psion Ltd 1984
Tous droits réservés

puis ARCHIVE est chargé.

XCHANGE sur IBM PC offre l'avantage de pouvoir passer d'un logiciel à l'autre très facilement sans changer de disquette. Ceci n'est pas possible sur le QL étant donné que les micro-cassettes n'ont pas une capacité suffisante pour stocker les quatre en même temps. Vous trouverez en annexe le listing d'un programme qui permet de charger les logiciels successivement en ayant accès à un menu qui se charge de vous guider pas à pas lors de vos manipulations.

SUR L'IBM PC

Si vous ne disposez pas de disque dur :

Mettez l'ordinateur sous tension sans oublier d'introduire la disquette contenant le système d'exploitation (DOS) dans le lecteur A. Si votre système « démarre » automatiquement sur le disque dur, ne mettez rien dans le lecteur A.

Lorsque le DOS est chargé et que le prompt indique :

```
A>
```

Remplacez la disquette du DOS par la disquette-programme que vous avez créée précédemment, insérez la disquette étiquetée « source » dans le lecteur B puis tapez :

XCHANGE <RETURN>

Après quelques instants le message suivant apparaît à l'écran :

XCHANGE
Version x.xx
Copyright Psion Ltd. 1984
Tous droits réservés

Vous êtes alors rapidement invité à indiquer le lecteur où se trouve la disquette « source » (la disquette originale qui sert à reconnaître que vous ne vous servez pas d'une copie pirate), tapez alors B et au bout d'une seconde environ, apparaît le menu principal.

RETIREZ IMMÉDIATEMENT LA DISQUETTE « SOURCE » ET REMPLACEZ-LA PAR UNE DISQUETTE FORMATÉE.

Si vous disposez d'un disque dur (unité C) :
Si lors de l'étape précédente vous avez installé XCHANGE sur le disque dur et que votre système démarre automatiquement sur ce dernier, il suffit, lorsque le prompt apparaît :

C>

de taper :

XCHANGE <RETURN>

sans oublier de mettre la disquette « source » dans le lecteur A et de l'indiquer lorsque le programme vous invite à le faire.

Modification pour faire exécuter XCHANGE au démarrage

Vous pouvez aussi modifier le fichier AUTOEXEC.BAT, qui se trouve avec le DOS pour lancer l'exécution automatique de XCHANGE lors du démarrage (réintialisation ou mise sous tension). Pour ce faire, il faut modifier AUTOEXEC.BAT à l'aide de EDLIN :

EDLIN AUTOEXEC.BAT

aussitôt « fin de fichier » apparaît à l'écran ; cela signifie que tout le fichier AUTOEXEC.BAT a été chargé

en mémoire vive. Vous pouvez visualiser les différentes lignes qui composent ce dernier en tapant L (pour liste) ; le fichier AUTOEXEC.BAT contient la plupart du temps KEYBFR (pour KEYboard FRENCH) et aussi WDATIM qui permet de mettre l'horloge interne à l'heure lors de la mise sous tension de l'appareil. AUTOEXEC.BAT peut aussi contenir d'autres commandes, cela dépend exclusivement de la façon dont vous l'avez configuré.

Pour lancer l'exécution de XCHANGE automatiquement il suffit d'ajouter une ligne de la manière suivante:

i (pour « insert »)

XCHANGE (pour indiquer le nom du programme que vous voulez voir exécuter automatiquement)

<CTRL> Z (pour arrêter l'insertion)

e (pour indiquer la fin de la modification)

Ceci fait AUTOEXEC.BAT est prêt et vous pouvez tester la modification en réinitialisant votre PC (<CTRL> <ALT> <DEL>). Après un court instant le chargement démarre automatiquement (vous voyez d'ailleurs progressivement s'afficher les différentes étapes) et la seule chose qu'il vous reste à faire est d'indiquer, le moment venu, dans quelle unité se trouve la disquette-source.

Maintenant que nous avons vu comment installer XCHANGE/ARCHIVE sur les deux machines, voyons comment utiliser le logiciel en lui-même.

CHAPITRE 2

Prise en main

2.1. PRÉSENTATION

Comme nous l'avons précisé plus haut XCHANGE sur IBM PC propose tout d'abord de sélectionner des tâches à partir d'un menu principal. Sur le QL, en revanche, à moins d'utiliser le programme proposé, chaque logiciel doit être chargé séparément sans passer par un menu. Le présent guide ne s'attachant qu'à ARCHIVE, nous nous placerons dans la situation où l'on se trouve directement dans cette application. Sur l'IBM PC, lorsque vous voyez apparaître le menu, sélectionnez "ARCHIVE — NOUV. TACHE" à l'aide de n'importe quelle touche et appuyez sur <RETURN>, vous vous trouvez alors dans ARCHIVE.

Remarquez la partition de l'écran selon différentes zones identiques pour tous les logiciels de la famille (EASEL, QUILL et ABACUS) ; notons cependant la possibilité de les modifier :

```
mode 0 <RETURN>
```

Immédiatement les zones que l'on distinguait précédemment disparaissent et il n'en reste plus qu'une seule, matérialisée par l'écran tout entier ; si vous tapez maintenant :

```
mode 1 <RETURN>
```

l'affichage revient comme auparavant.

SUR LE QL

L'instruction mode permet en outre de modifier le nombre de caractères par ligne ainsi que le nombre de couleurs disponibles :

mode 1,4 <ENTREE>

sélectionne un mode où les trois zones sont différenciées (mode **1**,4), mais où l'affichage est de 40 (mode 1,**4**) caractères par ligne et le nombre de couleurs de 8 au lieu de 4.

mode 0,6 <ENTREE>

sélectionne un mode où les trois zones sont confondues (mode **0**,6), et où l'affichage se fait sur 60 caractères par ligne (mode 0,**6**).

Le mode normal, si vous utilisez un moniteur est :

mode 1,8

En résumé, mode peut être suivi de 1 ou 0 pour distinguer ou confondre les zones d'affichage, puis (séparé par une virgule) de 4,6 ou 8 pour spécifier le nombre de caractères par ligne.

La meilleure façon pour se rendre compte des effets de cette commande est de faire un essai.

Vous pouvez aussi faire disparaître la partie supérieure (appelée « guide ») ; pour cela il suffit d'appuyer sur F2 ; vous disposez ainsi d'une surface de travail plus importante. Pour faire réapparaître le « guide », appuyez une nouvelle fois sur F2.

Cette zone montre de façon permanente les commandes disponibles sous ARCHIVE :

1) F1 — En appuyant sur F1, vous avez la possibilité de faire appel à un écran d'aide ; attention sur l'IBM PC, si vous avez installé XCHANGE sur une disquette, rappelez-vous que les fichiers d'aide se trouvent sur une seconde disquette qu'il faudra alors insérer lorsque vous en aurez besoin. Sur le QL ce fichier se trouve systématiquement sur la micro-cassette. Pour revenir où vous en étiez tapez plusieurs fois sur <ESC>.

2) F2 — Nous l'avons déjà abordé précédemment ; F2 vous

permet de disposer d'une surface de travail plus importante, en faisant disparaître le guide. Pour la faire réapparaitre, tapez une nouvelle fois sur F2.

3) F3 — Cette touche vous permet de visualiser sous la forme de quatre volets différents les différentes commandes disponibles sous ARCHIVE ; elles seront détaillées dans les prochains chapitres.

4) F6/ESC — Sur l'IBM PC cette touche de fonction (F6) vous permet de quitter momentanément une tâche pour revenir au menu principal (vous pourrez y revenir postérieurement en la sélectionnant). Sur le QL cet emplacement est noté "ANNULER — ESC" et sert à vous rappeler que toute commande peut être interrompue en tapant sur "ESC" ; cela est aussi vrai pour l'IBM PC.

5) Cette zone contient les volets des commandes qui peuvent être visualisées en appuyant plusieurs fois sur F3, comme précédemment indiqué.

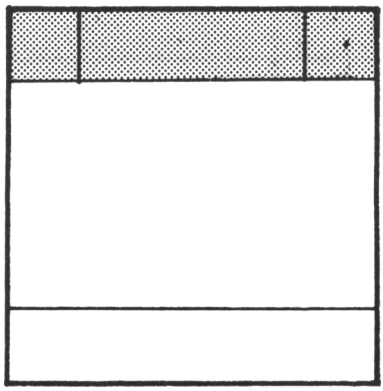


Figure 2.1
La zone de contrôle

* *la zone d'affichage*

C'est la zone où apparaissent les messages que vous validez, c'est aussi là que sont affichés les fichiers lorsqu'ils sont ouverts.

Vous pouvez par exemple taper la commande suivante :

```
écris "bonjour" <ENTREE>
```

Aussitôt le mot "bonjour" apparaît dans la zone d'affichage.

* *la zone de travail*

C'est la zone où s'inscrivent les commandes que vous tapez au

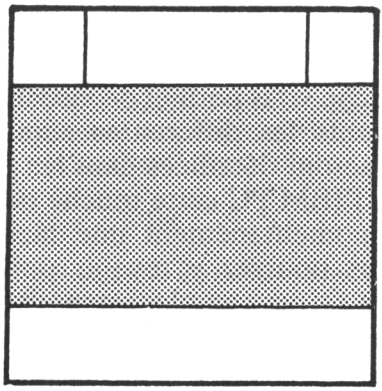


Figure 2.2
La zone d'affichage

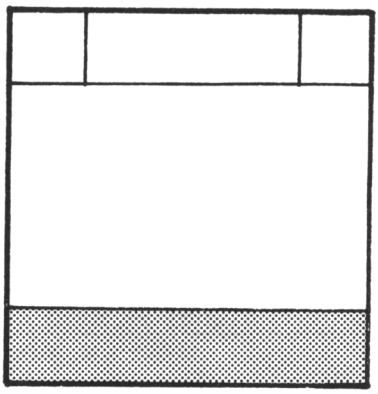


Figure 2.3
La zone de travail

clavier ainsi que les messages d'erreur. Par exemple si nous reprenons l'exemple précédent et que vous omettez un guillemet :

écris "bonjour <ENTREE>

un message d'erreur sera généré :

<Erreur 50 : guillemets de fermeture manquants

Notez la présence d'un numéro associé à l'erreur, celui-ci sera utile ultérieurement.

2.2. L'ÉDITEUR

Lorsque vous faites une faute de saisie (un caractère au lieu d'un autre par exemple), ARCHIVE dispose de commandes spéciales qui permettent de corriger la ligne que vous êtes en train de taper. Ces commandes sont légèrement différentes sur le QL et l'IBM PC. Nous allons les voir en détail :

IBM PC	QL	EFFET PRODUIT
← ou →	← ou →	Déplacement sur la ligne
<HOME> ou <END>	↑ ou ↓	Place au début ou en fin de ligne
<MAJUS> ←	<SHIFT> ←	Déplacement de mot en mot à gauche
<MAJUS> →	<SHIFT> →	Déplacement de mot en mot à droite
<DEL>	<CTRL> →	Effacement d'un caractère au prompt
←	<CTRL> ←	Effacement d'un caractère à gauche du prompt
<MAJUS> <END>	<CTRL> ↓	Effacement de la partie à droite du prompt
<MAJUS> <HOME>	<CTRL> ↑	Effacement de la partie à gauche du prompt

2.3. PRISE DE CONTACT

2.3.1. Les commandes à connaître absolument

Comme nous l'avons déjà dit, la manipulation d'ARCHIVE nécessite de connaître quelques commandes.

Il suffit de garder à l'esprit que rien n'est évident pour l'ordinateur et qu'il faut lui préciser toutes les actions qu'on veut le

voir effectuer ; ainsi "ouvre la boîte contenant le fichier-client" devra lui être spécifié, de même que "affiche la première fiche" et que "ferme la boîte".

Nous venons de voir que la touche F3 permettait de visualiser les diverses commandes disponibles ; **écris**, par exemple, fait partie de ces commandes. Il y en a environ une trentaine à connaître pour pouvoir manipuler un fichier facilement. Ces mots vont être listés dans les pages qui suivent avec un exemple de l'effet qu'ils produisent ; si vous ne comprenez pas tout immédiatement, contentez-vous de noter leur existence, nous les utiliserons en détail au cours des prochains chapitres.

2.3.2. Création d'un fichier

crée

Syntaxe :

```
crée <nfp> logique <nfl>
<champs-1>
<champs-2>
.........
fcrée
```

Fonction :

Permet de créer un fichier, en définissant les champs qui vont recevoir les informations à recueillir. Ne pas oublier de faire suivre le nom des champs par un "$" s'il est destiné à recevoir du texte.

Exemple :

```
crée "clients"
nom$
prénom$
No_compte
adresse$
code_postal
ville$
fcrée
```

Remarque : notez bien la présence du souligné entre No_compte et code_postal ; il ne doit pas y avoir d'espace entre deux mots dans un champ.

Lorsque l'on tape :

```
crée <RETURN>
```

Le prompt vous invite à rentrer un nom de fichier (chaque fichier doit avoir un nom distinctif afin de pouvoir se retrouver

ultérieurement). Ceci fait, l'utilisateur doit rentrer les libellés des différents champs (ils doivent être validés à chaque fois par <RETURN>, qui viennent alors s'afficher verticalement dans la zone du bas (zone de contrôle).

Lorsque tous les champs ont été définis, tapez une nouvelle fois sur <RETURN> et le fichier est créé ; il est prêt à recevoir des informations grâce à la commande **insère**.

N.B. lors de sa création, un fichier peut être suivi d'un nom de fichier logique. Ceci est utile lorsque l'on désire ouvrir plusieurs fichiers en même temps (ex. : le fichier "clients" et le fichier "fournisseurs"). Nous verrons ceci dans le chapitre III.

2.3.3. Insertion, modifications, suppressions

insère

Syntaxe : insère <RETURN>

Fonction :

Permet d'insérer des informations dans un fichier précédemment créé. Si ce dernier n'a pas été trié auparavant, les nouveaux enregistrements viendront s'insérer après le dernier enregistrement effectué (ou comme premier enregistrement si rien n'a encore été rentré). En revanche, si le fichier est classé en ordre (avec la commande **trie**), le nouvel enregistrement viendra s'insérer automatiquement à sa place.

Exemple : insère <RETURN>

Aussitôt après avoir tapé cette commande, un second prompt vient se positionner juste au début du premier champ qui a été défini, vous invitant ainsi à rentrer la première information.

A chaque fois que vous tapez sur <RETURN>, le prompt passe au champ suivant. Arrivé au dernier, le fait de taper sur <RETURN> une nouvelle fois enregistre définitivement la fiche. Si vous désirez revenir sur un champ (lorsque vous avez fait une faute par exemple), il suffit de taper sur <TAB> ou <SHIFT> <TAB> pour "descendre" ou "monter" les champs. Vous pouvez valider la fiche à n'importe quel moment en tapant sur <F5> ou bien abandonner l'opération d'insertion (et supprimer l'enregistrement que vous êtes en train de faire) en tapant sur <F4>.

<RETURN>	passe de champ en champ et valide l'enregistrement.
<TAB>	passe de champ en champ (du haut vers le bas) sans valider l'enregistrement.
<SHIFT> <TAB>	passe de champ en champ (du bas vers le haut), sans valider l'enregistrement.
<F5>	valide l'enregistrement quelle que soit la position du prompt.
<F4>	abandonne l'opération d'insertion et ne prend pas en compte l'enregistrement en cours.

modifie
Syntaxe : modifie <RETURN>
Fonction :

Permet de modifier les informations rentrées sur une fiche. Avant de modifier un enregistrement, il faut que celui-ci soit présent à l'écran (on l'appelle alors *enregistrement courant)*. Il peut être appelé avec les commandes **cherche, trouve, suiv, préc, pos, situe.**

Après avoir tapé:

modifie <RETURN>

Le prompt se place au début du premier champ : vous pouvez dès lors modifier l'information qui correspond à cette première rubrique grâce aux commandes de l'éditeur expliquées précédemment. Comme avec "insère", vous pouvez vous déplacer de champ en champ avec <RETURN> <TAB> ou bien <SHIFT> <TAB>, puis — la modification terminée — sur <F5> ; l'enregistrement est alors validé dans sa nouvelle forme et reste affiché à l'écran. Si vous tapez sur <F4>, vous abandonnez le mode "modifie" et vous annulez toutes les modifications qui auraient pu être faites.

efface
Syntaxe : efface <nfl>
Fonction :

Permet de "déchirer" une fiche. Il faut tout d'abord que

l'enregistrement que l'on veut supprimer soit présent à l'écran (qu'il soit courant) ; il suffit alors de taper :

efface <RETURN>

et l'enregistrement disparaît ; il est éliminé du fichier et remplacé par le suivant.

2.3.4. Opération de recherche

cherche :
Syntaxe : cherche "expression"
Fonction :

Permet de retrouver un enregistrement dans tout le fichier en donnant un critère de recherche spécifique.

Exemple : cherche nom$="Dupont" <RETURN>

Lorsque l'on valide cette opération, le fichier est lu du début jusqu'à la fin ; si une fiche répond au critère de recherche (ici nom$="Dupont"), la recherche s'interrompt et l'enregistrement devient courant. Vous pouvez bien entendu continuer la recherche (voir s'il y a d'autres enregistrements qui répondent à ce critère) en tapant :

continue <RETURN>

la recherche se prolongera alors à partir de l'enregistrement déjà détecté jusqu'à la fin du fichier ou bien jusqu'au nouvel enregistrement répondant au critère.

Vous pouvez bien évidemment faire votre recherche sur un autre champ :

Exemple : cherche code_postal=75001

et visualiser ainsi toutes les fiches des personnes habitant dans le premier arrondissement de Paris.

Remarque : notez que l'expression à rechercher doit être écrite entre guillemets ("") lorsqu'il s'agit de rechercher du texte (un nom ou une adresse par exemple), c'est-à-dire lorsque le champ est suivi du signe "$" et que dans l'autre cas (lorsque l'on recherche des chiffres comme un code postal par exemple), l'expression à rechercher ne doit pas être entourée de guillemets.

Notez aussi que "cherche" fait la différence entre les majuscules et les minuscules ; ainsi "DUPONT" n'est pas égal à "Dupont". Une recherche est donc susceptible de ne pas aboutir si les lettres ne correspondent pas exactement.

trouve

Syntaxe : trouve "expression"

Fonction :

permet de retrouver un enregistrement dans tout le fichier, dont un des champs (quel qu'il soit) répond au critère de recherche.

Exemple : trouve "Jean" <RETURN>

La recherche s'effectue du début jusqu'à la fin du fichier ou jusqu'à ce qu'une zone contienne l'expression recherchée. Dans l'exemple présent, la recherche s'arrêtera si un des champs contient le mot "Jean", que cela soit dans la rubrique des noms, des prénoms, de l'adresse ou bien de n'importe quel autre champ.

Remarque : la commande "trouve" ne fait pas la différence entre les majuscules et les minuscules ; ainsi "JEAN" est égal à "Jean".

Comme précédemment vous pouvez continuer la recherche à partir du premier enregistrement détecté en tapant :

continue <RETURN>

RAPIDEMENT...

En résumé, "cherche" et "trouve" permettent tous les deux de retrouver des enregistrements ; voici leurs principales différences :

cherche	trouve
— recherche un enregistrement sur un champ spécifique,	— recherche une expression dans tous les champs
— fait la différence entre les majuscules et les minuscules,	— ne fait pas la différence entre les majuscules et les minuscules,
— Ex. : cherche ville$ ="LYON"	— Ex. : trouve "boulevard"

SUR LE QL

pos
Syntaxe : pos "expression numérique"
Fonction :

Permet d'appeler une fiche par son numéro d'enregistrement ; chaque fiche dispose en effet d'un numéro d'enregistrement que vous pouvez visualiser en tapant :

écris numenr() <RETURN>

Si par conséquent vous tapez :

pos 5 <RETURN>

vous visualiserez l'enregistrement qui porte le numéro 5.

début
Syntaxe : début <nfl>
Fonction :

permet d'appeler le premier enregistrement du fichier, il devient à son tour l'enregistrement courant.

Exemple : début <RETURN>

suiv
Syntaxe : suiv <nfl>
Fonction :

Permet de faire défiler les enregistrements un à un du premier au dernier chaque "fiche" apparaissant sur l'écran devient l'enregistrement courant.

préc
Syntaxe : préc <nfl>
Fonction :

permet de faire défiler les enregistrements un à un, du dernier au premier. Chaque "fiche" apparaissant à l'écran devient l'enregistrement courant.

Exemple : préc <RETURN>

situe
Syntaxe : situe "expression"
Fonction :

si le fichier a été trié auparavant, **situe** permet de repérer directement une fiche répondant à un critère.

Exemple : situe "Dupont" <RETURN>

En supposant que le fichier ait été trié par ordre alphabétique croissant (sur le champ des noms), **situe** va aller directement vers les fiches où les noms commencent par "D" (il est en effet inutile de chercher depuis le départ puisque les noms sont rangés par ordre alphabétique), puis va rechercher à partir de cette position la fiche "Dupont".

Remarque : situe est beaucoup plus rapide que "trouve" ou "cherche".

continue
Syntaxe : continue
Fonction :

cette commande permet de continuer une recherche lancée avec "trouve" ou "cherche". La recherche continue à partir du dernier enregistrement trouvé jusqu'au prochain ou jusqu'à la fin du fichier si aucune fiche ne correspond plus au critère de recherche.

2.3.5. Ouverture et fermeture d'un fichier

ARCHIVE offre la possibilité de se servir de plusieurs fichiers en même temps. Comme nous l'avons déjà dit précédemment, pour passer d'un fichier à un autre il faut les désigner par leur nom logique qui vient s'ajouter au nom physique. Ce nom est facultatif lorsque vous n'ouvrez qu'un fichier à la fois, mais il devient obligatoire lorsque deux ou plusieurs fichiers sont ouverts en même temps.

Nous verrons plus en détail au cours des prochains chapitres comment il est possible de "jongler" avec les fichiers, mais il est bon de connaître l'existence des noms logiques dès à présent.

ouvre
Syntaxe : ouvre "nom physique" logique "nom logique"
Fonction :

permet d'ouvrir un fichier déjà créé et de pouvoir faire des modifications, des insertions, des suppressions, etc.

Exemple : ouvre "clients" logique "c"
ouvre "fournisseurs"

CONSEIL

Vous le remarquerez rapidement : lorsque plusieurs fichiers sont ouverts en même temps, il est nécessaire de faire appel aux noms logiques très souvent. Il est par conséquent plus commode et rapide de choisir des noms logiques le plus court possible, vous éviterez ainsi de perdre du temps en frappe.

Ex. : ouvre "fournisseurs" logique "f"
ouvre "carnet" logique "c"

Remarque : pour ouvrir un fichier, vous pouvez procéder de deux façons :

ouvre <RETURN> clients <RETURN>

ou bien

ouvre "clients" <RETURN>

lis

Syntaxe : lis "nom physique" logique <"nom logique">

Fonction :

Permet de lire un fichier uniquement pour la consultation. En d'autres termes, il n'est pas possible de faire une quelconque modification lorsqu'un fichier est ouvert avec "lis".

Exemple : lis "fournisseurs" logiques "f"
lis "clients"

De même qu'avec "ouvre", vous pouvez tapez la ligne directement suivie de <RETURN> ou "lis" puis <RETURN> et le nom du fichier.

ferme

Syntaxe : ferme "nom logique"

Fonction :

permet de fermer un fichier. Si plusieurs fichiers sont ouverts en même temps, il est possible de n'en refermer qu'un seul à la fois en précisant après "ferme", le nom logique du fichier que l'on désire clore. Si aucun nom n'est précisé, les fichiers sont refermés dans l'ordre où ils ont été ouverts.

2.3.6. Sélection et tri

isole
Syntaxe : isole <expression>
Fonction :

permet de séparer momentanément certaines "fiches" qui répondent à un critère donné.

Exemple : isole code_postal=45000

qui extrait momentanément toutes les "fiches" dont le code postal est égal à 45000. Vous pouvez visualiser la quantité de "fiches" restant après les avoir isolées en tapant :

écris nombenr() <RETURN>

Remarque : le fait d'isoler certains enregistrements ne signifie pas que les autres ont disparu, ils sont temporairement écartés, mais restent toujours en "mémoire".

restaure
Syntaxe : restaure <RETURN>
Fonction :

Permet de réintégrer tous les enregistrements qui avaient précédemment été écartés avec la commande **isole**. Si un critère de tri avait été déterminé, les enregistrements seront réintroduits selon ce critère.

tri
Syntaxe : trie champs;a/d
Fonction :

permet de trier un fichier par ordre croissant ou décroissant. Pour ce faire l'utilisateur doit déterminer un certain nombre de critères de tri (appelées "clés") qu'ARCHIVE suivra pour effectuer son classement.

Exemple : trie nom$;a <RETURN>

effectue un classement alphabétique des fiches sur les noms et dans un ordre ascendant (a→z).

Le classement peut bien entendu se faire aussi sur des chiffres.

Exemple : trie code-postal;d <RETURN>

effectue un classement des fiches sur les codes postaux et dans un ordre descendant (95→1). Ceci peut s'avérer extrêmement utile pour classer les envois PTT en nombre par département.

Deux ou plusieurs critères (clés) de tri peuvent être combinés.

Exemple : trie code-postal;a,nom$;a <RETURN>

effectue un tri ascendant, d'abord sur les codes postaux (plus petit vers plus grand), puis à l'intérieur des départements un tri des noms par ordre alphabétique. Dans ce cas, le tri va classer d'abord les fiches par département puis les noms par ordre alphabétique.

2.3.7. Communications entre les logiciels et copie de fichiers

Sur l'IBM PC, le transfert d'une tâche vers une autre s'effectue d'une façon extrêmement simple ; sur le QL, ce transfert n'est pas plus compliqué mais nécessite quelques manipulations. Dans un cas comme dans l'autre deux commandes sont accessibles (**exporte** et **importe**) et permettent de transférer facilement les fichiers vers les autres logiciels : EASEL, QUILL et ABACUS.

SUR L'IBM PC

Pour transférer rapidement un fichier ARCHIVE vers une autre application, il suffit de taper une fois sur <F6>, ce qui a pour effet de "geler" momentanément la tâche dans laquelle vous travaillez et de vous ramener au menu principal. A ce stade le fait de taper à nouveau sur <F6>, permet d'exporter les données d'ARCHIVE vers une autre tâche.

Attention : pour que cela soit possible, il faut que la tâche vers laquelle on veut exporter les données soit déjà ouverte. Dans le cas contraire ARCHIVE signalera qu'il n'y a pas de tâche ouverte et refusera d'exporter un fichier.

importe
Syntaxe : importe <nfp-orig> (logique <nfl>)
Fonction :

permet de transférer les données d'une tâche vers ARCHIVE. Pour ce faire, le fichier d'origine (celui que l'on veut "importer" doit d'abord avoir été créé (par exemple plusieurs rangées du tableur ABACUS), il porte alors l'extension .EXP ou .EXP. Le fichier de destination (celui qui sera importé) aura l'extension .DBF ou .DBF.

Exemple : importe "tableau1" en "archtab"

exporte
Syntaxe : exporte <nfp-dest>(;<var>) * (,<var>)*(quill)
Fonction :

Exporte le fichier en cours vers une autre application. Il est possible de n'exporter que certaines zones du fichier en cours en listant leurs noms après la commande. Par défaut, toutes les zones sont exportées.

Lorsque l'on désire exporter des données vers QUILL, il faut le préciser en fin de commande.

Exemple : exporte « clients » quill

copie
Syntaxe : copie <nfp-orig> en <nfp-dest>
Fonction :

Permet de faire une copie d'un fichier. Comme nous l'avons signalé précédemment, il est préférable de faire une copie de vos fichiers au moins en deux exemplaires pour ne pas risquer de perdre vos données. Faire une copie ne prend pas beaucoup de temps et peut vous en faire économiser beaucoup si une erreur de manipulation entraîne la perte de tout ce que vous avez rentré.

Exemple : copie "clients_dbf" en "clients2_dbf"

fait une copie du fichier "clients" sur le lecteur par défaut.

N.B. : lorsque vous désirez faire une copie d'un fichier, quel qu'il soit, vous devez *impérativement* préciser l'extension (voir ci-dessous).

LES EXTENSIONS

Lorsqu'un fichier est écrit sur une disquette ou une cassette, il est complété de trois lettres (en plus du nom de fichier) que l'on appelle **extensions**. Ainsi les fichiers construits grâce à la commande **crée** portent-ils l'extension. **DBF** (sur l'IBM PC) ou bien **_DBF** (sur le QL). Chaque sorte de fichier créé à partir d'ARCHIVE porte une extension distincte ; elles sont indiquées en annexe.

Exemple : clients_dbf (QL) ou bien clients.dbf (IBM PC)

Pour faire une copie sur un autre lecteur, il faut procéder de la façon indiquée ci-après.

SUR L'IBM PC

Retirez la disquette contenant le(s) logiciel(s) XCHANGE et remplacez-la par une disquette formatée. Tapez alors :

copie "b:clients.dbf" en "a:clients.dbf"

en supposant que le nom du fichier que vous voulez copier s'appelle « clients » et vous voulez le copier sur l'unité A avec le même nom.

Lorsque l'opération est terminée, n'oubliez pas de replacer la disquette contenant ARCHIVE à sa place (le lecteur A).

Si vous possédez un disque dur, vous pouvez procéder de la même façon, il suffit de préciser le nom du lecteur sur lequel vous voulez effectuer la copie avant le nom du fichier.

SUR LE QL

Retirez la micro-cassette contenant ARCHIVE et remplacez-la par une micro-cassette formatée. Tapez alors :

copie "mdv2_clients_dbf" en "mdv1_clients_dbf"

Ceci fait, remplacez à nouveau la micro-cassette de gauche par la copie d'ARCHIVE.

CONSEIL

Les copies de fichiers peuvent être relativement longues, le transfert se faisant par l'intermédiaire de la mémoire vive. Pour accélérer le processus, il suffit, lorsque la session est terminée, de revenir au Basic et de faire la copie à ce moment ; la mémoire vive n'étant plus encombrée par ARCHIVE, la copie s'effectuera beaucoup plus rapidement.

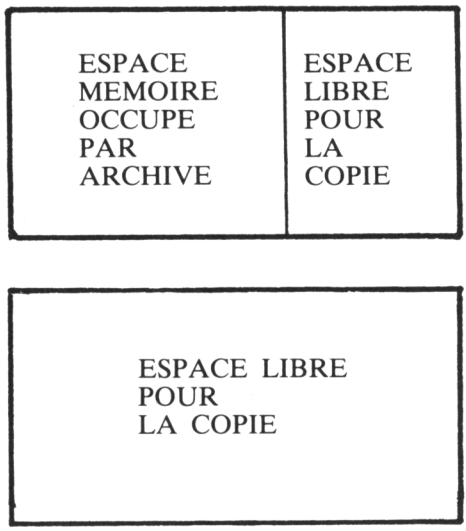


Figure 2.4
La copie de fichiers

Notez que sur l'IBM PC, **ajoute** permet de fusionner deux fichiers en un seul. Les deux fichiers doivent bien sûr avoir la même structure et dans le même ordre. Pour fusionner deux fichiers entre eux, il faut lancer la commande suivante :

```
copie "fichiers1" en "fichier2" ajoute
```

et *fichier2* recevra en plus des siens, les enregistrements de *fichier1*.

2.3.8. Les autres commandes

Les commandes ci-dessous abordent déjà le domaine de la programmation dans ARCHIVE ; elles permettent de réaliser des tâches automatiquement et de manière répétitive, évitant ainsi de lourdes manipulations.

partout/fpartout
Syntaxe : partout:action1*action2*:fpartout
Fonction :

partout signifie : *sur l'ensemble du fichier*. Grâce à cette commande (ou plutôt « instruction »), il est possible de faire exécuter par la machine une ou plusieurs opérations sur tous les enregistrements du fichier.

Exemple : partout:écris nom$:écris prénom$:fpartout <RETURN>

Une fois cette ligne validée, les noms et prénoms de tous les enregistrements du fichier en cours vont s'afficher à l'écran successivement. Notez la présence des deux points « : » entre chacun des ordres. Il s'agit d'une convention ; en ARCHIVE (comme en Basic) toutes les commandes écrites sur une même ligne doivent être séparées par deux points. Notez aussi « fpartout » à la fin de la ligne ; ceci signifie « fin de partout » et est obligatoire pour l'exécution d'une telle ligne : tous les ordres que l'on veut voir exécuter sur l'ensemble du fichier doivent être « pris en sandwich » entre « partout » et « fpartout ».

change
Syntaxe : change
Fonction :

permet d'introduire une modification dans un fichier (comme le ferait *modifie*) mais de façon répétitive et automatique. Supposons par exemple que l'on ait fait une faute sur les fiches des personnes qui habitent dans le premier arrondissement de Paris et qu'au lieu de 75001, on ait tapé 7501. Il va falloir retrouver toutes les fiches sur lesquelles on a fait l'erreur, les modifier, puis les valider à nouveau. Avec la commande « change » ce genre de modification va être extrêmement rapide et facile.

Il faut d'abord « isoler » tous les enregistrements où l'on a fait la faute :

```
isole code_postal=7501
```

Il faut ensuite donner l'ordre de la correction sur l'ensemble du fichier :

```
partout:que code_postal=75001:change:fpartout
```

Après quelques secondes, les corrections sont effectuées ; vous pouvez d'ailleurs le constater en utilisant les commandes **suiv** et **préc** pour visualiser quelques enregistrements. N'oubliez pas de taper :

```
restaure <RETURN>
```

Pour réintégrer tous les enregistrements ensemble.

Notez la ligne « **que code_postal=75001** », cela signifie « que le champ *code_postal* soit maintenant égal à 75001 » ; le mot **que** s'appelle une *instruction d'affectation*, nous l'utiliserons à chaque fois que nous aurons besoin d'affecter une variable.

N.B. : cette instruction correspond au **LET** en Basic, qui est généralement facultatif ; dans ARCHIVE elle est indispensable à toute affectation.

Il est possible d'effectuer les corrections que nous venons de voir d'une manière plus simple encore mais cela fait appel à une technique de programmation plus avancée que nous verrons dans le chapitre III.

état

Syntaxe : état*;champs1,champs2*

Fonction :

permet d'imprimer les champs qui composent un fichier suivies des informations qu'elles contiennent. Les données viennent s'imprimer selon un format fixe et prédéterminé (tabulaire). Vous pouvez, si vous le désirez limiter le nombre de champs à imprimer en le spécifiant après **état**. Si vous ne spécifiez rien, tous ces derniers sont imprimés.

Exemple :

```
état ; nom$,prénom$,adresse$
état
```

CONSEIL

Lorsque l'on emploie **état** tout le fichier commence par être lu pour connaître la longueur maximale des informations contenues dans chaque champ. En effet ceci détermine la présentation de l'impression. Dans le cas où la longueur des informations excède le

nombre de caractères imprimables par ligne (généralement 80), l'imprimante saute automatiquement à la ligne suivante et les informations sont mélangées et illisibles. Faites donc attention à ce que la longueur maximale des informations contenues dans les champs dont vous demandez l'**état** n'excède pas la capacité de votre imprimante.

active
Syntaxe : active "nom logique"
Fonction :

Lorsque plusieurs fichiers sont utilisés en même temps, « active » permet de spécifier lequel d'entre eux doit être utilisé. Pour cela, il faut qu'ils aient été ouverts avec un *nom logique* (obligatoire si plusieurs fichiers le sont en même temps).

Exemple : active "c"

Fait du fichier dont le nom logique est "c", le fichier courant.

ajoute
Syntaxe : ajoute <nlf>
Fonction :

permet d'ajouter des enregistrements à un fichier autrement que par **insère**. Cette commande fonctionne de la même façon que **change** ; c'est-à-dire qu'elle s'emploie au sein d'un programme ou d'une mise à jour automatique.

Exemple : que ville$="LYON":ajoute

créera un enregistrement où le champ *ville*$ contiendra le mot "LYON".

2.3.9. Les accès à la mémoire de masse

cat
Syntaxe : cat *<lecteur>*
Fonction :

cette commande liste à l'écran tous les fichiers présents sur une disquette ou une cassette. Si aucun nom de lecteur n'est précisé entre les guillemets, c'est le catalogue du lecteur par défaut qui est affiché.

Exemple :

```
cat "a:" (IBM PC)
cat "mdv1_" (QL)
cat "" <RETURN>
```

tue

Syntaxe : tue "nom_de_fichier_extension"

Fonction :

permet de supprimer un fichier de la mémoire de masse. Attention, cette action est irrémédiable ; un fichier **tué** ne peut être récupéré.

Exemples :

```
tue "b:fichier_dbf"
tue "mdv2_fichier_dbf"
```

N.B. : L'extension du fichier que vous désirez supprimer doit toujours être spécifiée explicitement.

SUR L'IBM PC

Des commandes supplémentaires sont disponibles sur l'IBM PC :

index

qui permet de visualiser les différentes clés de tri qui ont déjà été utilisées. Dans tous les cas il existe un index 0, qui est l'ordre historique, c'est-à-dire celui dans lequel les informations ont été rentrées la première fois. Vous pouvez avoir d'autres index (un maximum de 5) si par exemple vous triez les enregistrements par ordre alphabétique sur les noms puis par les prénoms. Dans ce cas vous pourrez disposer très facilement de trois index en tapant la commande suivante :

active index 1

qui provoquera immédiatement un classement par ordre alphabétique sur les noms (index 1) sans que vous ayez à repasser par la commande « tri ». Puis, si

vous le désirez, vous pouvez revenir à l'ordre historique en tapant la commande :

active index 0

qui toujours immédiatement vous remet le fichier dans l'état initial.

renomme

qui permet de *renommer* un fichier.

Exemple : renomme "fichier_dbf" en "fich_dbf"

noms

qui affiche à l'écran tous les noms logiques des fichiers ouverts.

RAPIDEMENT :

voici un résumé très rapide des commandes que nous avons vues :

crée	crée "clients" logique "c"	création d'un fichier
insère	insère	insertion de nouveaux enregistrements
modifie	modifie	modification manuelle de l'enregistrement courant
efface	efface	suppression de l'enregistrement courant
cherche	cherche nom$="Dupont"	recherche une expression dans un champ particulier
trouve	trouve "Jean"	recherche une expression dans tous les champs
début	début	fait du premier enregistrement l'enregistrement courant
fin	fin	fait du dernier enregistrement l'enregistrement courant
suiv	suiv	défile les enregistrements du début vers la fin
préc	préc	défile les enregistrements de la fin vers le début
situe	situe "Martin"	positionne à l'endroit où figure une expression au moins égale à l'expression recherchée (le fichier doit avoir été trié au préalable)
continue	continue	permet de continuer une recherche au-delà du premier enregistrement trouvé

ouvre	ouvre "clients" logique "c"	permet d'ouvrir un fichier déjà créé (avec la possibilité de faire des modifications)
lis	lis "clients" logique "c"	permet d'ouvrir un fichier déjà créé (sans possibilité de faire des modifications)
ferme	ferme "c"	referme un fichier
isole	isole N°_compte<1000	isole momentanément certain enregistrements selon un critère
restaure	restaure	réintègre les enregistrements isolés avec « isole »
trie	trie nom$;a,prénom$;a	trie les enregistrements selon le contenu des champs spécifiés
importe	importe "c1" dans "c12"	importe un fichier ABACUS ou EASEL et fait un fichier ARCHIVE
exporte	exporte "c12";nom$	exporte un fichier ARCHIVE vers ABACUS ou EASEL, ou QUILL si cela est précisé à la fin de la commande
copie	copie "b:bd" en "a:bd2" copie "mdv1_bd" en "mdv2_bd2"	copie intégralement un fichier
partout/ fpartout	partout :écris nom$:fpartout	effectue une action sur l'ensemble du fichier
change	que nom$="Martin":change	change le contenu d'un champ automatiquement
état	état;nom$,prénom$	provoque l'impression du contenu de tout ou d'une partie d'un fichier
active	active "f"	fait du fichier dont le nom logique est spécifié, le fichier courant
affiche	affiche	provoque l'affichage du contenu d'un fichier à l'écran
ajoute	que nom$="Martin":ajoute	ajoute un enregistrement automatiquement
cat	cat ""	liste le contenu de la mémoire de masse
tue	tue "Fichier_dbf"	supprime un fichier de la mémoire de masse

Commandes spécifiques au QL:

pos	pos 5	recherche un enregistrement par son numéro d'enregistrement

Commandes spécifiques à l'IBM PC:

index	index	affiche l'état des index en cours
active index	active index 1	permet d'activer un index en l'appelant par son numéro
efface index	efface index	permet de libérer un index de tri.

2.4. CRÉATION PRATIQUE D'UN FICHIER

Dans cette partie, nous allons suivre pas à pas la création d'un fichier dans son intégralité. Tout d'abord quelques rappels :

Un fichier est composé de champs (autant de champs que de rubriques sont nécessaires — par exemple nom, prénom, etc.)

Il est indispensable de déterminer au préalable si une zone est destinée à recevoir du texte ou des chiffres (si elle est suivi d'un "$" ou non).

Nous allons réaliser un fichier très simple dans lequel figurera simplement le nom, le prénom et le numéro de téléphone des personnes que nous désirons mettre dans notre fichier. Pour cela, il nous faut trois champs : un pour le nom (texte), un pour le prénom (texte), et un pour le téléphone (chiffres).

Assurez-vous qu'une disquette ou micro-cassette formatée se trouve dans l'unité prise par défaut (B : ou mdv2_).

2.4.1. Création des champs du fichier

Tapez d'abord :

crée <RETURN>

Tapez ensuite le nom du fichier (ex. : « carnet »).

Tapez une nouvelle fois sur <RETURN> ; le nom du fichier est enregistré, prêt à être structuré.

Vous pouvez maintenant entrer les champs du fichier un à un en n'oubliant pas de taper sur <RETURN> après chacun d'entre eux. N'oubliez pas non plus le "$" pour les champs qui contiennent du texte.

Lorsque vous aurez terminé, tapez une nouvelle fois sur <RETURN> et le fichier sera définitivement créé.

VOICI LES OPÉRATIONS QUE VOUS AURIEZ DÛ EFFECTUER :

crée <RETURN>

carnet	<RETURN>
nom$	<RETURN>
prénom$	<RETURN>
tél	<RETURN>
	<RETURN>

Maintenant que le fichier est créé, vous pouvez visualiser sa structure à l'écran en tapant :

affiche <RETURN>

2.4.2. Remplissage du fichier

A présent, vous pouvez commencer à enregistrer les informations qu'il va contenir (les noms, prénoms, et numéros de téléphone). Pour ce faire, tapez :

insère <RETURN>

Aussitôt la structure du fichier apparaît à nouveau à l'écran — notez le mot "maître" à côté du champ "logique" — et le prompt se place directement à côté du champ "nom$" vous invitant ainsi à taper le premier nom. Ceci fait, passez au champ suivant en tapant sur <TAB> ou <RETURN> et tapez le prénom ; effectuez la même opération pour le numéro de téléphone (attention rappelez vous que vous ne pouvez taper que des chiffres). Maintenant si vous tapez à nouveau sur <RETURN> l'enregistrement est validé, il disparaît de l'écran laissant ainsi la place à un nouvel enregistrement. Si vous tapez sur <TAB>, le prompt repasse au premier champ (nom$), vous permettant ainsi de faire une correction à l'aide des commandes expliquées à la section 2.4.

Quelle que soit la position du prompt, il est toujours possible de valider un enregistrement en appuyant sur la touche <F5>.

EN RÉSUMÉ :

champ suivant	<TAB> ou <RETURN>
Validation	<F5> ou <RETURN> si le prompt est sur le dernier champ
Champ en champ	<TAB> ou <SHIFT> <TAB>

Une fois la validation d'un enregistrement effectuée, l'écran se vide des informations et le prompt se positionne à nouveau à côté

du premier champ, vous invitant à rentrer un second enregistrement.

Répétez l'opération, autant de fois que vous désirez rentrer des enregistrements. Lorsque vous avez terminé n'oubliez pas de valider le dernier (l'écran ne doit plus montrer que les champs du fichier sans qu'aucune information n'y figure plus).

Pour arrêter l'insertion, tapez :

<ESC> ou <F4>

Le dernier enregistrement réapparaît alors et l'opération est maintenant terminée. A ce stade vous pouvez continuer avec le fichier que vous avez créé ou bien le refermer pour vous en servir ultérieurement, et dans ce cas taper :

ferme <RETURN>

Le lecteur de droite se met en marche pendant un court instant et le fichier est refermé. Il est indispensable, lorsque l'on veut arrêter d'utiliser un fichier, de toujours le refermer. Si ceci n'était pas fait, le fichier serait inutilisable une seconde fois.

Sachez cependant que si vous tapez :

"quitte" <RETURN>

c'est-à-dire abandon d'ARCHIVE, le fichier est automatiquement refermé.

Sur IBM PC, "Quitte" supprime la **tâche** mais pas le **fichier** ; cela signifie que lorsque vous revenez au menu principal XCHANGE, la tâche ARCHIVE dans laquelle vous travaillez aura disparu mais pas le fichier que vous aurez créé. Pour le retrouver il vous faudra retourner dans ARCHIVE en nommant une nouvelle tâche et en ouvrant le fichier que vous désirez utiliser.

2.5. UTILISATION DE QUELQUES COMMANDES SIMPLES

A l'aide du fichier que nous venons de créer, nous allons voir comment utiliser quelques commandes simples. Pour que les exemples que nous allons montrer soient probants, il faut que le nombre d'enregistrements soit suffisant et que vous ayez entré les

mêmes noms que ceux sur lesquels nous allons montrer les exemples. Voici donc une série de noms que vous devrez intégrer à notre base de données (rentrez-les tels qu'ils sont, y compris les **fautes d'orthographe**.

Durand	Paul	42447099
Pierre	Philippe	25508733
Amand	Silvie	45464321
Auclair	Silvie	55558799
Chenesseau	élène	47890336
Durand	Jean	98970723
Monnod	Anne	56453423
d'Arquey	antoine	88897654
Dasseux	Pascal	56789076
Dupont	Monique	67898000

Pour rajouter des noms, il suffit de taper à nouveau **insère** et de procéder comme la première fois ; les noms viendront s'inscrire après le dernier enregistrement effectué.

N.B. : si précédemment vous avez refermé le fichier, il faut naturellement le réouvrir avant de rentrer de nouveaux enregistrements :

ouvre "carnet" <RETURN>

Commencez par vérifier les extrémités (le premier et le dernier enregistrement) en tapant :

début <RETURN>

et

fin <RETURN>

Si rien n'apparaît à l'écran, il faut « forcer » l'affichage en tapant :

affiche <RETURN>

et recommencer **début** et **fin** ; si le dernier enregistrement ne correspond pas à "Dupont Monique", c'est que vous avez oublié de le valider — vous avez tapé trop tôt sur **<F4>** — vous devrez donc le réintroduire en tapant à nouveau **insère** et en n'oubliant pas de le valider avec **<F5>** (l'écran doit être vide lorsque vous tapez sur **<F4>** pour quitter le mode **insère**.

Vous pouvez aussi visualiser le nombre d'enregistrements que vous avez effectués en tapant :

écris nombenr() <RETURN>

Nous avons dit dans le premier chapitre que l'instruction **écris** servait à afficher à l'écran n'importe quel message entre guillemets. Ici **nombenr( )** n'est pas entre guillemets parce qu'il s'agit d'une *variable* (un nom qui contient automatiquement le nombre d'enregistrements rentrés en machine) ; on demande d'afficher le contenu d'une variable sans la mettre entre guillemets. Vous pouvez vous-même définir vos variables et les « remplir » avec vos propres valeurs.

Exemple : a, b, c, d, numéro, x, y, indice

Tous ces mots sont des noms de variable (elles peuvent être définies sur un maximum de 13 caractères mais un seul suffit). De même que pour les champs, la convention veut que les variables qui sont destinées à recevoir des chiffres ne soient suivies d'aucun signe et que celles qui sont destinées à recevoir du texte soient suivies du signe distinctif "$".

Exemple : lettre$, mot$, a$, b$, x$

sont des noms de variable destinées à recevoir des lettres (on les appelle des variables *alphanumériques* par opposition à numériques).

En fait, les champs et les variables sont tout à fait comparables ; la seule différence qui existe entre les deux est tout simplement qu'un champ fait partie d'un fichier, alors qu'une variable n'existe que temporairement, *jusqu'à ce que l'ordinateur soit étient.*

Vous pouvez vous même « affecter » vos propres variables en donnant l'ordre suivant :

que <nom de variable>=<valeur> <RETURN>

ou bien

que <nom de variable$>=<"texte"> <RETURN>

Exemples : que indice=15 <RETURN>
que mot$=« voici une chaîne de caractères » <RETURN>

Lorsque vous validez ces lignes, les variables sont *affectées* des

valeurs suivant le signe "=". Le mot **que** s'appelle **une instruction d'affectation** et signifie grossièrement :

que la variable soit maintenant **égale** à telle valeur

Vous pouvez d'ailleurs visualiser le contenu d'une valeur que vous venez d'affecter en tapant la commande suivante :

```
écris indice <ENTER>
écris mot$ <ENTER>
```

Tout à fait logiquement, il existe des noms de variables qui sont interdits ; ce sont les noms qu'ARCHIVE interprète directement comme étant des mots qu'il connaît déjà. C'est le cas des noms de commandes, par exemple (écris, lis, affiche, etc.) ; ces mots sont dit « réservés » puisqu'ils sont définis à la base comme significatifs. Vous ne pouvez pas écrire :

```
que écris=5
```

puisque "écris" est déjà répertorié comme commande donnant l'ordre à l'ordinateur d'afficher des données à l'écran. Il existe une centaine de mots réservés qui correspondent à des mots qu'ARCHIVE interprète déjà.

Vous pouvez aussi, pour visualiser le contenu du fichier, faire circuler les fiches une à une à l'écran en utilisant les commandes **suiv**, qui montre successivement les fiches une à une à l'écran de la première vers la dernière, et **préc** qui fait la même chose mais dans l'autre sens.

N'oubliez pas que chaque commande doit être validée avec la touche :

```
<RETURN>
```

Cela doit être devenu maintenant instinctif et nous ne l'indiquerons plus nécessairement dorénavant.

2.5.1. Recherche d'un enregistrement

Dans le fichier que nous avons créé figurent par fiche (enregistrement) un nom, un prénom et un numéro de téléphone. Il peut être intéressant de retrouver dans un premier temps un enregistrement par son nom :

Exemple : `cherche nom$="Durand"`

Dans notre fichier, deux personnes portent le nom de

"Durand", ARCHIVE va donc s'arrêter une première fois sur le premier enregistrement qui correspond à ce critère et il suffira de taper :

```
continue
```

pour voir apparaître la seconde fiche libellée au même nom, et ainsi de suite jusqu'à ce qu'ARCHIVE atteigne la fin du fichier sans découvrir d'autre personne portant ce nom.

ÉCONOMISER DU TEMPS ET DE LA FRAPPE...

Toutes les commandes rentrées par l'intermédiaire du clavier sont gardées en mémoire (tampon) tant que l'on n'en tape pas une nouvelle. Ainsi lorsque l'on doit utiliser une commande de manière répétitive plusieurs fois de suite, il est inutile de la retaper à chaque fois... Il suffit simplement d'appuyer sur <**F5**>, qui a pour effet de faire réapparaître la dernière commande entrée au clavier.

Cela peut s'avérer très utile pour les commandes telles que **continue, suiv préc**, etc. qui doivent normalement être tapées successivement lorsque l'on recherche des enregistrements particuliers. Cette touche est aussi utile lorsque l'on tape une commande d'une longueur importante et que l'on a fait une faute de syntaxe qu'ARCHIVE refuse ; il suffit de rappeler la ligne que l'on vient de taper grâce à <**F5**>, de la corriger avec l'éditeur, puis de la valider à nouveau avec <RETURN>.

Il existe une autre variable qui permet de savoir si une recherche a été fructueuse ou non (si une fiche portant le nom recherché existe), elle est un peu comme **nombenr()** que nous avons vu précédemment et porte le nom de **vu()**. Ainsi :

```
écris vu()
```

affichera la valeur 1, si la recherche a été couronnée de succès et 0 si aucune fiche ne correspond au critère recherché.

Si l'on veut faire une recherche immédiate de "Paul Durand", il suffit de préciser outre le nom, le prénom :

```
cherche nom$="Durand" et prénom$="Paul"
```

Et presque immédiatement l'enregistrement recherché apparaît (rappelez-vous que vous devez forcer l'affichage avec la

commande **affiche** si rien n'apparaît). Il aurait bien sûr été possible de spécifier d'autres critères de recherche en plus de ceux utilisés, mais la simplicité du fichier ne nécessitait pas un affinement si précis.

Attention, souvenez vous qu'avec **cherche** les majuscules et les minuscules sont différenciées, il faut donc prendre soin de bien taper "Durand" avec un "D" majuscule autrement ARCHIVE chercherait en vain une correspondance.

Maintenant, supposons par exemple que l'on ne se souvienne plus si "Monnod" s'écrit avec un ou deux « n ». Il suffit alors de taper la commande suivante :

```
trouve "Mon"
```

L'ordinateur va rechercher dans tous les champs s'il existe une suite de lettres semblable et va tout d'abord s'arrêter sur l'enregistrement portant le nom « Dupont », à cause du prénom « **Mon**ique », qui contient bel et bien la suite de caractères « Mon ». Il suffira alors de taper **continue** pour que la recherche se poursuive et que l'on trouve enfin l'enregistrement "Monnod". Bien entendu rechercher une fiche avec **trouve** est bien plus long puisqu'ARCHIVE doit rechercher dans tous les champs de la chaîne de caractères indiquée — ce qui n'est pas le cas avec **cherche** qui ne vérifie que les champs spécifiés. Une autre particularité de **trouve** est qu'il ne fait pas la différence entre les majuscules et les minuscules, ce qui peut s'avérer extrêmement utile lorsque l'on ne connaît pas exactement la façon dont est écrite une chaîne recherchée.

Une dernière méthode pour repérer un enregistrement est la commande **situe**. Pour pouvoir l'utiliser, le fichier doit avoir été préalablement trié. **Situe** offre l'avantage d'être la méthode la plus rapide pour rechercher une fiche. Contrairement aux autres, cette commande trouve toujours un enregistrement sur lequel s'arrêter même s'il n'y a pas correspondance parfaite.

Exemple :

```
trie nom$;a
situe "Dasseux"
```

vous pouvez aussi spécifier deux critères de localisation si votre fichier a été trié sur deux champs :

Exemple :

```
trie nom$;a,prénom$;a
situe "Dasseux", "Pascal"
```

Rien ne vous oblige à « situer » un nom existant ; vous pouvez par exemple essayer :

```
trie nom$;a
situe "z"
```

Qui aura pour effet de s'arrêter sur le nom commençant par la lettre la plus proche de la fin de l'alphabet (z).

2.5.2. Sélection d'un « paquet de fiches »

Il est possible de sélectionner momentanément quelques fiches qui correspondent à un critère donné. Ceci est surtout utile lorsque l'on manipule un long fichier, l'ensemble est alors moins long à gérer.

Exemple : `isole nom$="Durand"`

Seules les deux fiches contenant le nom "Durand" seront gardées temporairement dans le fichier. Vous pouvez le vérifier en demandant à ARCHIVE le nombre d'enregistrements restant :

```
écris nombenr()
```

ce qui donne 2.

Pour réintégrer toutes les fiches ensemble, tapez **restaure**. Toutes les fiches sont de toute façon réintégrées lorsque vous refermez un fichier.

2.5.3. Tri, ordonnancement

Une des possibilités simples et très puissantes d'ARCHIVE est de pouvoir trier des enregistrements selon n'importe quel critère et surtout selon plusieurs critères en même temps. Nous pouvons par exemple trier le fichier que nous avons constitué par ordre alphabétique sur les noms (ce que vous avez peut être déjà fait si vous avez essayé la commande **situe**) :

```
trie nom$;a
```

Le « a » après le point-virgule signifie *ordre ascendant*, on peut aussi trier par ordre descendant, il suffit pour cela de taper un « d » à la place du « a ». Après quelques instants le fichier est trié par ordre alphabétique (sur les noms).

SUR LE QL

Le fichier restera classé par ordre alphabétique si vous l'avez ouvert avec la commande **ouvre** (si vous ne l'avez ouvert que pour le consulter, c'est-à-dire avec la commande **lis** il ne sera que temporairement trié).

SUR L'IBM PC

ARCHIVE dispose d'une commande supplémentaire ; en effet il est possible de définir jusqu'à quatre tris différents et de les conserver en mémoire jusqu'à ce que vous décidiez de les supprimer. Cette commande vous permet dans un premier temps de vérifier tous les tris disponibles, de les activer et de les supprimer.

index affiche les différents tris effectués jusqu'alors ; celui qui est actif au moment ou vous tapez **index** est précédé su signe « = »

active index vous donne la possibilité de choisir un tri parmi ceux qui ont déjà été opérés vous évitant ainsi de repasser par la procédure de tri ; ceci est utile lorsque vous avez par exemple besoin alternativement d'un classement par ordre alphabétique puis par code postal.

efface index vous permet de supprimer un index lorsque vous juger inutile de garder un tri dont vous n'avez plus besoin. Pour cela, il faut d'abord *activer* l'index que l'on veut supprimer avec **active index** puis taper **efface index**. *Vous ne pouvez pas supprimer un index autrement* (par exemple en tapant efface index 3).

Il faut garder à l'esprit deux choses en ce qui concerne les index :

l'index 0 ne peut être supprimé ; c'est l'index qui contient *l'ordre historique*, c'est-à-dire l'ordre dans lequel ont été rentrés les enregistrements la première fois.

L'index 5 est réservé à l'*isolement* momentané de certaines fiches comme nous l'avons vu précédemment. Ainsi en tapant **active index 5** certaines fiches se trouvent immédiatement isolées (si vous l'avez déjà fait) sans que vous soyez obligé de passer par le circuit habituel de l'isolement.

Vous pouvez demander non seulement que les noms soient classés par ordre alphabétique, mais aussi les prénoms (second critère de tri). Dans ce cas, si plusieurs fiches comportent le même nom, c'est le prénom qui les différenciera.

Ce tri ne présente pas un intérêt énorme : seules deux fiches

comportent le même nom (Durand) ; nous pouvons tout de même constater l'efficacité d'un tel classement : Durand Jean va bel et bien se trouver avant Durand Paul si vous tapez la commande suivante :

```
trie nom$;a,prénom$;a
```

Nous aurions, bien sûr, pu classer aussi le fichier par les numéros de téléphone (c'est-à-dire sur des chiffres) mais cela ne présente pas non plus un intérêt particulier. Cela pourrait en avoir un si nous disposions d'un champ contenant des chiffres d'affaires par exemple.

2.5.4. Modifications manuelles

Si vous ne l'avez pas rectifié vous-même lors de la saisie des noms listés plus haut, vous avez pu constater que les prénoms comportaient quelques erreurs (majuscules oubliées, lettres manquantes, etc.). Ceci était fait à dessein pour illustrer la possibilité de pouvoir modifier très simplement un enregistrement sur lequel on aurait fait des fautes. Par exemple le prénom Hélène est amputé de son « H ». Pour pouvoir le remplacer, il faut d'abord retrouver l'enregistrement :

```
cherche prénom="élène"
```

puis lorsque la fiche apparaît tapez :

```
modifie
```

Aussitôt le prompt se place à côté du premier champ (nom$), vous invitant à faire une modification. Dans le cas présent, ce n'est pas le premier champ qui nous intéresse mais le second. Pour y passer, il suffit de taper sur <RETURN> ou <TAB> comme cela fut expliqué auparavant, puis d'actionner les touches → ou ← pour désactiver la suppression automatique (voir encadré), revenir au début du mot et taper un « H ». Validez ensuite l'enregistrement corrigé avec <F5> et l'opération est terminée ; l'enregistrement reste affiché à l'écran, mais le prompt repasse dans la zone de contrôle.

LA SUPPRESSION AUTOMATIQUE

Lorsque l'on fait une modification sur un enregistrement, ARCHIVE vous permet de supprimer automa-

tiquement le contenu d'un champ sans que vous ayez à effacer la ligne entière au préalable. Lorsque le prompt se place **pour la première fois, au début d'un champ**, si vous tapez une autre touche que →, ←, <RETURN>, <TAB> ou une touche de fonction, c'est-à-dire si vous tapez une lettre ou un chiffre, la ligne existante est entièrement supprimée au profit des nouvelles données que vous voulez rentrer. En conséquence, si vous ne voulez pas voir disparaître la ligne entière et effectuer une correction sur la **première lettre du champ**, il vous faut d'abord actionner les touches → et ← pour désactiver l'effacement automatique.

Nous pouvons procéder de la même façon pour corriger le prénom qui n'a pas de majuscule :

```
cherche prénom$="antoine"
```

Dès que l'enregistrement, apparaît tapez à nouveau **modifie** et faites la correction comme précédemment en vous servant de l'éditeur.

La dernière opération consiste à remplacer le « i » de « Silvie » par un « y ». Nous allons le faire grâce à des commandes plus sophistiquées.

2.5.5. Modifications automatiques

ARCHIVE permet encore de faire des modifications sur l'ensemble du fichier de manière automatique, évitant ainsi de passer toutes les fiches en revue et de taper **modifie** à chaque fois, ce qui peut être long surtout si les mêmes fautes ont été commises dans plusieurs enregistrements différents.

La commande **partout/fpartout**, que vous avez déjà vue, va nous aider dans la réalisation de ces tâches répétitives.

Nous allons donc modifier l'enregistrement où « Sylvie » est mal orthographié ; dans le cas présent, il y a deux enregistrements où ce prénom est mal orthographié. Il suffit donc de demander à ARCHIVE de parcourir le fichier du début jusqu'à la fin **(partout)**, de regarder **si** « Sylvie » est mal orthographié, de le remplacer par la bonne orthographe **(change)** et de terminer l'opération **(fpartout)**.

En conséquence, la commande devra être la suivante :

```
partout :si prénom$="Silvie" :que prénom$="Sylvie"
:change : fsi :fpartout
```

Voici en quelques mots comment il est possible de modifier un fichier entier sans avoir à passer toutes les fiches en revue. Vous pouvez le vérifier, les enregistrements auront bien été corrigés. Le mot **si** que nous n'avons pas encore vu jusqu'ici, s'appelle un *test* et a pour fonction d'effectuer les ordres qui suivent **si** une condition est vérifiée — de même que **partout/fpartout**, les actions à exécuter doivent être « prises en sandwich » entre **si** et **fsi**. **Change** est l'instruction qui permet de faire des modifications de manière automatique ; si nous avions mis **modifie** à la place, l'exécution se serait arrêtée et le prompt serait venu se placer devant le premier champ, attendant une modification *manuelle* comme nous l'avons vu précédemment.

2.6. UTILISATION DE QUELQUES COMMANDES PLUS COMPLEXES

2.6.1. La notion de fonction

Vous aurez pu le constater, tous les noms que nous avons rentrés sont (à part l'initiale) écrits en minuscules, ce qui n'est pas très heureux. Pour y remédier, nous allons « transformer » tous les noms en majuscules. Et ceci grâce à une **fonction** dont dispose ARCHIVE et qui va permettre de réaliser ceci de manière très simple : **majus()**. Cette fonction transforme n'importe quel texte figurant entre les parenthèses en majuscule ; par conséquent :

```
partout:que nom$=majus(nom$):change:fpartout
```

N.B. : Les initiales qui sont déjà en majuscules ne sont pas affectées par l'opération que nous venons de réaliser ; elles sont en majuscules et elles le restent.

Vous pouvez encore une fois vérifier que tout a bien été effectué comme nous le souhaitions en visualisant quelques enregistrements.

Puisqu'il existe une fonction qui transforme les minuscules en majuscules, il y a bien entendu une autre fonction qui produit l'effet inverse. Il s'agit de **minus()** qui fonctionne exactement de la même façon. Si nous décidions de l'utiliser pour remettre les noms dans l'état précédant nous transformerions alors *toutes* les lettres en minuscules (y compris les initiales) ce qui n'est guère satisfaisant. Pour éviter cela ARCHIVE offre la possibilité de ne traiter qu'une partie des lettres d'un mot ; en effet toutes les lettres qui composent ce mot vont être repérables par leur *position* dans

ce dernier. Ainsi pour "Durand", « D » est en première position, « u » en deuxième, « r » en troisième, etc. Pour isoler un caractère dans un texte, il suffit de spécifier après celui-ci (entre parenthèses) quelles sont les lettres que l'on veut isoler grâce à l'instruction **jusqu**.

Exemple : nom$(jusq 1) → premier caractère du contenu de la variable nom$
"texte"(jusq 3) → les trois premiers caractères de texte » soit « tex »

Vous pouvez aussi repérer un ensemble de lettres à l'intérieur d'un mot :

Exemple : "texte"(4 jusq 5) → les deux dernières lettres de la chaîne "texte" (te)

Une autre possibilité est de déterminer la longueur d'un mot avec la fonction **long** () qui donne la longueur (en chiffres) de la chaîne spécifiée entre parenthèses :

Exemple : écris long(" texte") → 5
écris long(nom$) → qui donne la longueur de la chaîne contenue dans nom$

Si, par conséquent, nous reprenons notre problème précédent, à savoir transformer en minuscules les noms de famille mais pas leurs initiales, il suffit de taper la commande suivante :

```
partout:que nom$=majus(nom$(2 jusq long(nom$))):chan
ge:fpartout
```

et le tour est joué ! Nous avons bien tranformé tous les noms, à partir de la seconde lettre et jusq'à la fin de chacun d'entre eux (jusqu'à leur longueur totale).

Vous voyez donc qu'un nombre quasiment infini de solutions à vos problèmes sont envisageables et cela dans un temps record (imaginez la même opération sur un fichier écrit à la main !). ARCHIVE dispose de nombreuses autres fonctions qui vous seront nécessaires si vous voulez entreprendre de créer de « véritables » applications utilisables en un minimum de temps et avec un maximum d'efficacité. Pour cela il faut connaître les autres fonctions et en particulier la possibilité de programmer ARCHIVE

pour automatiser le plus possible vos applications ; nous y reviendrons dans le prochain chapitre.

SAVOIR PRÉVOIR...

Si vous avez consulté quelques fiches après toutes les manipulations que nous avons opérées, vous aurez probablement remarqué que certains enregistrements posaient des problèmes. En premier lieu la fiche portant le nom « d'Arquey » dont le « d » a été transformé en majuscule alors que cela n'a pas lieu d'être. D'autre part, lorsque nous avons classé les enregistrements par ordre alphabétique sur les noms, « d'Arquey » est venu se positionner tout à fait à la fin, alors que la coutume veut que l'on fasse abstraction de la particule dans un cas comme celui-ci.

Pour y remédier (ce genre de choses sont assez courantes dans les fichiers, et l'on ne saurait rejouer la Nuit du 4 Août tous les dix ans !), nous vous conseillons deux solutions :

La première qui consiste à réserver un champ spécial destiné à recevoir les particules (on pourrait l'appeler *part$* et qui ne contiendrait que les « de », « du », « d », etc. Dans ce cas, il faudra bien prendre en compte ce champ lorsque l'on affichera les noms à l'écran où sur une imprimante.

La seconde évite de réserver un champ pour les particules et consiste simplement à intégrer ces dernières aux prénoms correspondants. Ainsi dans le champ *prénom$* nous aurions non pas seulement "Antoine" mais "Antoine d'" et dans le champ nom$ simplement "Arquey", ce qui ne poserait alors plus de problème.

Ces deux solutions sont envisageables ; le plus important lorsque vous choisissez d'en adopter une plutôt qu'une autre est de *prévoir* l'usage que vous ferez de votre fichier ; si les prénoms y doivent être affichés ou imprimés *systématiquement* alors la seconde solution est la meilleure (économie de temps), si ce n'est pas le cas, il vaut mieux perdre un peu de temps à saisir uniquement les particules pour vous assurer que les noms seront imprimés selon le format correct.

Le second problème porte sur les numéros de téléphone. Nous les avons rentrés dans un champ numérique puisqu'a priori ils ne sont composés que de chiffres. En fait, la plupart du temps ils sont soit séparés par un point (44.47.67.78), par un espace (43 56 78 90) ou bien par tout autre séparateur y compris les parenthèses pour indiquer l'indicatif parisien ou étranger (16 (1) 42.27.98.50).

On ne peut considérer dans ce cas, que les numéros de téléphone ne sont composés que de *chiffres*. Il est donc préférable de prévoir *dès la création du fichier* un champ *alphanumérique* (suivi du "$") pour ces derniers.

Ce n'est pas parce qu'un champ doit contenir des chiffres qu'on ne peut le déclarer alphanumérique lors de sa création.

C'est exactement la même chose pour les codes postaux ; la plupart des pays utilisent un système composé de chiffres uniquement, mais qu'arrivera-t-il le jour où vous devrez saisir le code postal d'un client anglais ou canadien pour lequel des lettres seront nécessaires ?

En outre, les champs alphanumériques permettent de traiter l'information de façon beaucoup plus souple ; nous avons vu précédemment que nous avions la possibilité de localiser certaines lettres par l'intermédiaire de leur emplacement dans une chaîne. Il est par conséquent beaucoup plus facile de sélectionner certaines fiches par département si l'on peut les repérer par les deux premiers « caractères » du champ *code_postal*. Notez que la solution qui consiste à traiter les chiffres est aussi possible, mais beaucoup plus compliquée.

Exemple : `isole code_postal$(1 à 2)="69"`

sélectionne les fiches dont le code postal commence par 69, autrement dit toutes les fiches de Lyon et du Rhône.

Pour arriver au même résultat sur un champ numérique, il faut traiter le code postal comme un nombre et par conséquent le diviser par 1 000 pour ne garder que les deux premiers chiffres :

```
isole code-postal/1000=69
```

N.B. : *Le signe « / » signifie « diviser »*

ce qui est considérablement plus long et plus lourd à traiter.

Tous ces « détails » sont à prendre en compte lorsque l'on crée un fichier et nous ne manquons pas de l'indiquer chaque fois qu'un problème de prévision peut se produire.

2.6.2. Les opérations

ARCHIVE, nous l'avons vu, dispose de fonctions qui permettent d'effectuer des opérations sur les chaînes. Ceci ne serait pas complet si les nombres ne pouvaient pas, eux aussi, être manipulés facilement.

En premier lieu, il faut connaître les opérateurs qui permettent de faire les opérations « classiques », puis ceux qui permettent de *comparer* deux expressions entre elles (ex. : plus grand que... plus petit que...) et enfin les opérateurs booléens qui permettent de traiter l'information logiquement.

Calcul arithmétique

+	addition
—	soustraction
/	division
*	multiplication
( )	parenthèses
^	élévation à une puissance

Notez les deux signes « / » et « * » pour les divisions et les multiplications, ils sont légèrement différents de ceux que l'on emploie habituellement.

Notez en outre que le signe qui caractérise les nombres décimaux n'est pas la virgule (,) mais le point (.), comme sur les machines à calculer.

Ex. : que x=3.4567

La comparaison

>	plus grand que	si a>20: écris "sup à 20":fsi
<	inférieur à	si a<20: écris "inf à 20":fsi
=	égal	si a=20: écris "égal à 20":fsi
>=	plus grand ou égal	si a>=20: écris "sup ou égal":fsi
<=	inférieur ou égal	a<:20: écris "inf ou égal":fsi
<>	différent de	si a<>20: écris "dif de 20":fsi

Chaque comparaison qui est vérifiée renvoie la valeur 1 et 0 lorsqu'elle est fausse. Ainsi :

écris a>20

affichera 1, à chaque fois que **a** sera supérieur à 20.

Les opérateurs logiques

et et logique — est vérifié si *deux conditions* sont vraies.

partout : si nom$="Durand" et prénom$="Jean" : écris "Trouvé":fsi: partout

affichera "Trouvé", uniquement si un enregistrement portant le nom de "Durand" et le prénom de "Jean" est détecté.

ou ou logique — est vérifié si l'une ou l'autre des conditions est vraie.

partout : si nom$="Durand" ou nom$="Dupont" : écris "Trouvé":fsi:fpartout
affichera "Trouvé" si un enregistrement contient le nom "Durand" ou "Dupont".

non non logique — est vérifié si l'expression qui suit **non** est différente de celle qui précède **non**.

partout : si non nom$="Durand" : écris "Différent de Durand":fsi: partout

affichera "Différent de Durand" si les enregistrements ne comportent pas le nom "Durand".

De même que pour les opérateurs de comparaison chaque opération logique qui est vérifiée renvoie la valeur 1 et 0 lorsqu'elle est fausse. Ainsi :'

```
écris non a=20
```

affichera 1, à chaque fois que a sera différent de 20.

Ces opérateurs sont d'une importance capitale dans le traitement des fichiers. Ils sont la base de toutes les sélections, tris et autres ordonnancements qui peuvent s'opérer sur un fichier. Grâce à eux, un fichier va pouvoir être scruté dans tous les sens et ceci de la manière la plus rapide qui soit.

Par exemple, si l'on dispose d'un fichier de clients où figurent, outre les coordonnées, le chiffre d'affaires réalisé avec eux, il sera possible de faire une liste de tout ceux qui sont vos meilleurs clients.

Exemple :

```
partout : si CA>"10000": montre:fsi:
fpartout
```

Cette simple ligne aura pour effet de lister à l'écran tous les clients avec qui le chiffre d'affaires est supérieur à 10 000 F.

Mais cela ne se limite pas uniquement à des opérations de ce genre ; il est naturellement possible de combiner plusieurs opérateurs entre eux, pour arriver à une sélection encore meilleure. Supposons par exemple que vous souhaitiez envoyer un publipostage à tous les clients qui ont à *la fois* un chiffre d'affaires supérieur à 10 000 F *et* qui résident à l'étranger.

Il vous suffira de lancer une commande de ce style pour voir s'imprimer tous les noms des personnes répondant à ces critères :

```
partout : si CA>10000 et PAYS$<>"FRANCE":écris
nom$:fsi:fpartout
```

et cela ne prendra que quelques instants.

Aucun doute : la puissance d'ARCHIVE vous réserve des possibilités considérables, nous vous en montrerons encore d'autres au cours des prochains chapitres, mais auparavant voyons les autres fonctions dont vous pouvez vous servir pour optimiser les services d'ARCHIVE.

2.6.3 Les autres fonctions

Nous avons vu dans la section précédente quelques fonctions et les opérateurs. Nous allons voir maintenant les fonctions mathématiques, puis les fonctions sur les chaînes, les variables système (qui sont apparentées à des fonctions), et les fonctions « spéciales ».

LES FONCTIONS...

Bien qu'il vous soit possible de « fabriquer » vous-même vos propres fonctions selon vos besoins, certaines opérations dont on a souvent besoin ont été intégrées à ARCHIVE.

On les appelle des fonctions **intrinsèques** ; cela signifie qu'elles font partie intégrante du logiciel et qu'elles existent de toute façon, que vous vous en serviez ou non. Il y a un nombre impressionnant de fonctions dans ARCHIVE que nous allons découvrir.

12 FONCTIONS MATHÉMATIQUES

Elles ont pour rôle de calculer le résultat d'une expression numérique. Par exemple la racine carrée d'un nombre, ou bien son sinus, etc. Elles sont

suivies de deux parenthèses au sein desquelles doit figurer le nombre que l'on veut faire « passer » dans la fonction. A l'exception de **pi()** qui donne le nombre pi, il faut obligatoirement qu'il y ait un argument entre parenthèses. Ne mettez pas d'espace entre le nom de la fonction et la parenthèse, il s'en suivrait une erreur de syntaxe.

abs()	valeur absolue d'un nombre	abs(- 1)	1
atg()	arc-tangente	atg(5)	1.3734
cos()	cosinus	cos(3)	- 0.9899
deg()	conversion en degrés un angle en radians	deg(1)	57.2957
ent()	donne la partie entière d'un nombre	ent(4.5)	4
exp()	exponentielle	exp(2)	7.3890
ln()	logarythme naturel	ln(6)	1.7917
pi()	donne la valeur de pi	pi()	3.1415
rac()	racine carrée d'un nombre	rac(25)	5
rd()	convertit en radian un nombre en degrés	rd(3)	0.0523
sin()	sinus	sin(3)	0.1411
tg()	tangente	tg(2)	- 2.1850

Pour toutes les fonctions qui suivent, nous avons indiqué, lorsque cela était possible, leur correspondance en Basic.

* les fonctions sur les chaînes.

long()
Syntaxe : long(exp alpha)
Fonction :

donne la longeur de l'expression alphanumérique entre parenthèses. La longueur d'une chaîne est le nombre de caractères qu'elle contient (y compris les espaces).

Exemple :
```
écris long("texte")
écris nom$(long(nom$) jusqu long
(nom$))
```

Le premier exemple donne la valeur 5, (nombre de caractères de la chaîne "texte") et le second affiche la dernière lettre de la chaîne contenue dans la variable nom$.

Emploi : cette fonction est surtout employée lorsque l'on veut

tester une lettre particulière dans une chaîne dont on ne connaît pas la longueur, à la manière du second exemple.

Basic : LEN

majus()
Syntaxe :
Fonction : majus(exp. alpha)

convertit en majuscules l'expression alphanumérique entre parenthèses. Si les caractères sont déjà en majuscules, cette fonction n'a aucun effet.

Exemple : que t$=majus("texte")

où t$ se voit affecté la chaîne "TEXTE".

Emploi : nous avons déjà donné un exemple de l'emploi de cette fonction dans la partie précédente. Elle peut s'avérer extrêmement utile pour unifier certains champs que l'on veut voir écrits de la même façon.

Basic :

minus()
Syntaxe :
Fonction : minus(exp. alpha)

Convertit en minuscules l'expression alphanumérique entre parenthèses. Si les caractères sont déjà en minuscules, cette fonction est sans effet.

Exemple : que t$=minus("TEXTE")

où t$ se voit affecté de la chaîne "texte".

Emploi : même emploi que la fonction précédente.

Basic :

repro()
Syntaxe :
Fonction : repro(exp. alpha,n)

créer une chaîne de **n** caractères, constituée par le premier caractère de l'expression alphanumérique en première position.

Exemple : repro("a",6)

créer une chaîne de six caractères, remplie de "a" ("aaaaaa")

repro("texte",2)

créer une chaîne de deux caractères, remplie de "t" (première lettre de la chaîne texte).

Emploi : cette fonction est précieuse lorsque l'on veut par exemple aligner des messages apparaissant à l'écran (ou sur une imprimante). On peut ainsi remplir une chaîne avec un nombre **n** de caractères blancs (espace) et s'arranger pour que toutes les chaînes que l'on veut voir s'afficher à l'écran aient exactement la même longueur. On peut aussi l'utiliser pour tracer des traits de séparation à l'écran :

repro("-",80)

fournira une ligne constituée de "-" sur 80 caractères.

Basic : REPT

souschn()
Syntaxe : souschn(chaîne1, chaîne2)
Fonction :

vérifie si la chaîne2 est présente dans la chaîne1. Si c'est le cas, **souschn()** renvoie la position (en chiffre) ou la *chaîne2* apparaît dans la *chaîne1*.

Exemple : écris souschn("Antoine","nt")

qui affichera à l'écran le chiffre 2, car la chaîne "nt" apparaît dans Antoine à partir de la deuxième position.

Emploi : cette fonction est très utilisée lorsque l'on recherche une partie de chaîne dans une autre. Le fait qu'elle renvoie une position autorise toutes sortes d'opérations. Enfin, dans les fichiers elle permet entre autres d'utiliser la commande **cherche** comme **trouve**.

cherche souschn(nom$,"urand")

Vous permet de rechercher *uniquement* dans le champ "nom$", la chaîne "urand" ; ce qui aurait aussi été possible avec **trouve** mais sur tous les champs.

N.B. : à la différence de **trouve, souschn** fait la différence entre les majuscules et les minuscules.

Basic : INSTR

* Les fonctions sur les dates et l'heure

La plupart de ces fonctions nécessitent que l'horloge interne de votre appareil ait été remise à l'heure lors de la mise sous tension.

date()
Syntaxe : date(n)
Fonction :

produit une chaîne contenant la date du jour. Selon la valeur de l'argument **n** le format de cette date est différent :

date(0) aaaa/mm/jj
date(1) jj/mm/aaaa
date(2) mm/jj/aaaa

où aaaa représente l'année, mm le mois et jj le jour.

Exemple : écris date(0)

qui donne par exemple 1986/01/01, si vous avez remis l'horloge interne de votre système à l'heure avant de charger ARCHIVE.

Emploi : cette fonction peut être utile, lorsque l'on utilise ARCHIVE pour enregistrer des commandes, par exemple ; il ne sera plus besoin de rentrer la date du jour mais simplement de spécifier une expression du genre :

que date$=date(1)

qui aura pour effet d'enregistrer le jour où aura été passée la commande automatiquement — nous l'emploierons dans nos exemples pratiques.

Basic : DATE$

heure()
Syntaxe :
Fonction : heure()

produit une chaîne qui contient l'heure sous le format HH:MM:SS, ou HH sont les heures, MM, les minutes et SS les secondes.

Exemples : écris heure()

N'oubliez pas les deux parenthèses après cette fonction, même si elles ne servent en fait à rien.

Emploi : on peut très bien imaginer des applications dans lesquelles on ait besoin de garder une trace de l'heure à laquelle ont été faits certains enregistrements, mais cela est rarissime.

Basic : TIME$

jours()
Syntaxe :
Fonction : jours (exp alpha)

renvoie le nombre de jours qui se sont écoulés depuis le 1er janvier 1583, date à laquelle a vraiment commencé l'application du calendrier grégorien tel que nous le connaissons aujourd'hui.

Exemple : écris jours(« 1986/01/01 »)

Notez bien le format dans lequel la date doit être donnée : aaaa/mm/jj.

Emploi : a priori, il peut paraître bizarre de prendre le 1er janvier 1583 comme date de référence. En fait, il suffit d'un peu de conversion pour ramener cela à une date plus proche. Cette fonction est tout à fait utilisable comme ceci ; par exemple, si vous désirez connaître le nombre de jours qui se sont écoulés entre une livraison et le paiement de la facture, il sufit de procéder comme suit :

```
que délai$=jours(datepaie$)-jours(dateliv$)
écris délai$
```

et l'écart entre les deux dates vient s'inscrire tout simplement. C'est une commande très puissante, car elle tient compte des années bissextiles et du nombre de jours par mois !

Basic : —

mois()
Syntaxe : mois(n)
Fonction :

produit une chaîne de caractères, qui contient le nom du mois dont le numéro est spécifié entre les parenthèses. Si **n** prend une valeur supérieure à 12, aucun message d'erreur n'est généré, et un résultat est toujours obtenu ; il correspond à **n** *modulo* 12, c'est-à-dire le reste de la division de **n** par 12.

Exemple : écris mois(1)

qui affiche « janvier »
Emploi : cette fonction est surtout utile pour produire des rapports de façon plus claire et précise, et surtout lorsque l'on ne dispose que d'un chiffre en guise de mois. Vous pouvez essayer la ligne suivante :

```
écris mois(date(0) (4 jusq 5))
```

qui aura pour effet d'afficher le nom du mois contenu dans l'horloge interne de votre système. Remarquez l'utilisation du découpage de la chaîne **date (0)**.

Basic : -

* *les variables système*

Les variables système sont celles qu'ARCHIVE s'est réservées pour contenir certaines valeurs dont il a besoin au cours du traitement d'un ou plusieurs fichiers.

nombenr()
Syntaxe : nombenr(!nlf!)
Fonction :

contient le nombre d'enregistrements d'un fichier ; si plusieurs fichiers sont ouverts en même temps, il est possible de spécifier

entre les parenthèses le *nom logique de fichier* sur lequel on veut obtenir l'information.

Exemple : écris nombenr(« c »)

affiche le nombre d'enregistrements contenus dans le fichier ouvert avec le nom logique "c".

Emploi : cette variable est indispensable à connaître ; nous nous en sommes déjà servi pour vérifier le nombre d'enregistrements présents dans le fichier que nous avions créé précédemment, et elle sera utile pour toutes les applications.

nombz()
Syntaxe : nombz(<n>!,<nlf>!)
Fonction :

produit une chaîne qui contient le nom du champ spécifié par son numéro d'ordre (le premier champ étant le n° 0, le second le n° 1 etc.). Si plusieurs fichiers sont ouverts en même temps, il est possible de spécifier un nom logique de fichier en supplément ; on obtient alors les noms des champs figurant dans le fichier désigné par son nom logique.

Exemple : écris nombz(0, "c")

donne le nom du premier champ du fichier ouvert avec le nom logique "c".

Emploi : l'emploi de cette fonction est très utile pour certaines applications particulières, lorsque l'on dispose d'une variable qui contient le numéro des champs d'un fichier.

Basic : -

numerr()
Syntaxe : numerr()
Fonction :

fournit un nombre qui correspond au numéro de la dernière erreur qui s'est produite lors de l'exécution d'un programme. Il existe un grand nombre d'erreurs possibles.

Exemple : écris numerr()

affiche le numéro de la dernière erreur si il y en a eu une ; dans le cas contraire, **numerr()** renvoie la valeur 0.

Emploi : voici encore une variable qu'il faut absolument connaître. Elle permet de constituer ce que l'on appelle une routine d'erreur. Lors de l'exécution d'un programme, si vous faites une erreur son exécution s'interrompt et vous devez la relancer une fois l'erreur corrigée. Avec une routine de traitement d'erreur, il est possible de signaler simplement que vous avez fait une erreur, de vous inviter à la corriger, puis de continuer l'exécution du programme comme si rien ne s'était passé. Cela ne vous semble peut-être pas très utile, mais vous le verrez avec la pratique, ce genre de routine est indispensable.

Basic : ERRNUM

typez()
Syntaxe : typez(n!,<nlf>!)
Fonction :

permet de tester le type d'un champ (savoir s'il est numérique ou alphanumérique). Toujours suivant le même principe, les champs sont répertoriés selon un numéro d'ordre où le premier champ porte le n° 0, le second le n° 1 et ainsi de suite. Si plusieurs fichiers sont ouverts en même temps, il est possible de spécifier *un nom logique de fichier*.

Si le champ est de type numérique la valeur renvoyée est 0 et 1 dans le cas contraire.

Exemple : écris typez(0)

renverra 1, si le premier champ du fichier courant est de type alphanumérique.

Emploi : cette fonction est utile pour contrôler si les saisies qu'un utilisateur peut faire sont valides.

si typez(0)=1:écris « ENTREZ DU TEXTE »:sinon:écris « ENTREZ UN NOMBRE »:fsi

ce qui provoquera l'affichage du message « ENTREZ DU TEXTE » si la fonction **typez()** détecte que le premier champ est alphanumérique, et « ENTREZ UN CHIFFRE » dans le cas contraire.

Basic : -

valeurz()
Syntaxe : valeurz(n!,<nlf>!)
Fonction :

fournit la valeur du champ désigné par **n** (0 pour le premier champ, 1 pour le deuxième, etc.). Si plusieurs fichiers sont ouverts en même temps, il est possible de spécifier un *nom logique de fichier*.

Emploi : cette fonction n'offre d'avantages que lorsque l'on dispose du *numéro d'ordre* des champs qui constituent un fichier.

* *les fonctions qui convertissent des valeurs en chaînes*

Nous avons signalé précédemment qu'un champ qui était destiné à recevoir des chiffres ne devait pas nécessairement être déclaré de type *numérique*. Nous avons dit que les chaînes offraient des possibilités de traitement bien plus ouvertes, notamment grâce à la facilité du *découpage* décrite auparavant. En fait, à l'aide des fonctions que nous allons voir maintenant, il est possible de convertir soit une chaîne de caractère en une valeur numérique soit une valeur numérique en chaîne.

LES CONVERSIONS

Comment est-il possible de transformer une chaîne de caractères en nombre, vous êtes-vous peut-être demandé ? En fait cette conversion n'est bien sûr possible que si ladite chaîne de caractères contient effectivement des chiffres :

Ex. : "123"

qui, en soi n'est que la suite de trois caractères, mais qui, une fois transformée en chiffres devient un nombre parfaitement exploitable par ARCHIVE. Souvenez-vous le signe + est le signe de la *concaténation*, c'est-à-dire le signe qui permet de « coller » deux chaînes entre elles ; mais c'est aussi le signe de l'addition lorsqu'il figure entre deux chiffres. Par conséquent, il ne faut pas confondre :

et "123" + "456" qui donne la chaîne "123456"
123 + 456 qui donne le résultat 579

Maintenant, grâce aux fonctions présentées ici, il

est possible de convertir les nombres en « lettres » et les lettres en « nombres » !

chn()
Syntaxe: chn(nombre,format,décimales)
Fonction :

convertit un **nombre** en chaîne selon un **format** adaptable et avec un nombre de **décimales** (nombre de chiffres après la virgule) ajustable.

format peut prendre quatre valeurs différentes selon la présentation que l'on désire :

0 — décimal (avec des chiffres après la virgule)
1 — exponentiel (pour les mathématiques)
2 — entier (arrondi, sans chiffre après la virgule)
3 — général (sans format particulier)

décimales peut prendre n'importe quelle valeur, selon le nombre de chiffres après la virgule que l'on veut voir apparaître. En général, deux sont suffisamment précis.

Exemple : écris chn(25.98764,2,2)

qui donnera 26, c'est-à-dire la partie à gauche de la virgule arrondie, étant donné que nous avons spécifié le format **2**. Notez qu'il a été obligatoire de donner un chiffre pour les décimales bien que par définition un nombre entier n'ait pas de chiffre après la virgule ; cela reste cependant obligatoire pour des raisons de syntaxe (ce chiffre est aussi obligatoire pour le format général).

écris chn(255.7654,0,2)

donne 255.76 puisque le format choisi est le n° 0 qui correspond au *décimal*, avec un nombre de chiffres après la virgule égal à 2.

Emploi : même si cette fonction vous paraît compliquée, en fait à l'usage elle est extrêmement utile surtout pour produire des affichages propres et correspondant tous au même format. Une somme en francs, par exemple, ne se calcule jamais avec plus de deux chiffres après la virgule (ce qui peut arriver si vous divisez 10 F en 3).

Basic : PRINT USING

deci()
Syntaxe : deci(nombre,décimales,longueur)
Fonction :

convertit un **nombre** en une chaîne de caractères avec un format *décimal*. La **longueur** de la chaîne peut être spécifiée ainsi que le nombre de **décimales** à prendre en compte. Remarquez que le **nombre** vient toujours s'inscrire à droite de la chaîne ainsi créée.

Exemple : écris deci(234.9876,2,12)

ce qui donne : " 234.98"

et si vous donnez l'ordre d'afficher un deuxième chiffre, cette fois moins long, il sera parfaitement aligné sur la droite du précédent :

écris deci(6.56,2,12)

ce qui donne : " 6.56"

Emploi : encore une fois, cette fonction permet de normaliser l'affichage de chiffres en les alignant tous de la même façon.

Basic : PRINT USING

gene()
Syntaxe : gene(nombre,longueur)
Fonction :

Agit exactement de la même façon que **deci()**, mais en convertissant un nombre en format général. Il n'est donc pas besoin de spécifier un nombre de chiffres après la virgule puisque le format général s'en occupe par lui-même. En revanche, il faut toujours préciser la longueur de la chaîne dans laquelle va s'écrire ce nombre.

Exemple : écris gene(123.456,12)

donne : " 123.456"

Emploi : il est le même que pour **deci()**, cela dépend du format que l'on désire.

val()
Syntaxe : val(exp. alpha)
Fonction :

convertit une chaîne en valeur. La chaîne doit contenir des chiffres en première position, sinon la valeur renvoyée est 0.

Exemple : écris val("123")
donne : 123
et : écris val("12étages")
donne : 12
en revanche : écris val("nbre étages : 12")
donne : 0

Emploi : cette fonction est indispensable à connaître ; il arrive très souvent que l'on ait à convertir une chaîne en chiffres.

Basic : VAL

valvar()
Syntaxe : valvar(exp. alpha)
Fonction :

cette fonction renvoie la valeur contenue par la chaîne entre parenthèses. Attention, la chaîne entre parenthèses doit correspondre au nom d'une variable contenant déjà une valeur, sinon un message d'erreur est généré.

Exemple : que nom$="tex"
que texte=20
écris valvar(nom$+"te")
donne 20

Que s'est-il passé ? Nous avons d'abord dit que la variable alphanumérique *nom$* était égale à "tex", ensuite nous avons donné une valeur à la variable numérique *texte* (20), puis nous avons reconstitué le nom de cette variable au sein de **valvar()** en concaténant ce que contenait *nom$* ("tex") et "te", ce qui reconstituait bien le nom de la variable *texte* ; il suffisait que cette dernière se trouvât dans la fonction **valvar()** pour que son contenu en fût extrait.

Emploi : cette fonction est extrêmement puissante, il est

primordial de bien l'assimiler ; nous nous en servirons, dans plusieurs applications.

* Les fonctions ASCII

ASCII

Ces cinq lettres sont les initiales d'**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange et doivent se prononcer "asqui". Ce code est donc international et la plupart des constructeurs de matériels et des concepteurs de logiciels (dont naturellement PSION, auteur d'ARCHIVE) s'y conforment. Chaque caractère est associé à un nombre et ceci dans la majorité des systèmes. Ainsi, la lettre "A" majuscule est associée à 65 et "a" minuscule à 97 (les minuscules se trouvent toujours 32 caractères plus loin que les majuscules).

Tout cela ne présenterait pas un intérêt énorme, pour autant que vous puissiez afficher la lettre "a" par l'intermédiaire du clavier, si ce fameux code ne renfermait pas des talents cachés. Ainsi les lettres conventionnelles se trouvent toutes sur le clavier mais le symbole "*1*" par exemple n'y figure pas. Dès lors le code ASCII va nous être utile, il suffit de repérer le numéro auquel il est associé et de le transformer en "*1*". C'est à cela que servent les instructions que nous montrons ici.

Il faut cependant noter que les claviers permettent de générer ces caractères spéciaux ; reportez-vous à votre manuel d'utilisateur.

En outre, les codes ASCII permettent d'envoyer des *codes de contrôle* à une imprimante. En effet, la plupart des imprimantes disposent de plusieurs polices de caractères (*italique*, **gras**, etc.) et seuls les codes ASCII permettent de les sélectionner. Ainsi, sur une imprimante EPSON FX 80, il faut envoyer le code **<ESC> "4"** pour sélectionner l'impression en italique. A partir d'ARCHIVE, il suffira donc d'envoyer le code correspondant à <ESC> suivi du chiffre "4" et le tour sera joué.

Les codes ASCII servent encore à beaucoup d'autres choses, dont nous n'avons pas besoin ici ; notons cependant la commodité des caractères graphiques. Ceux-ci n'existent que sur l'IBM PC et sont associés à des numéros qui se situent bien au-dessus des caractères normaux, et qui ont pour effet d'afficher des traits, ou des dessins à l'écran. Attention, ces codes ne sont pas nécessairement reconnus par votre imprimante.

L'ASCII SUR L'IBM PC

On peut obtenir les caractères ASCII par l'intermédiaire du clavier sur l'IBM PC. Pour cela il suffit de maintenir la touche <ALT> (Alternate) enfoncée, de taper le numéro de code correspondant au caractère que l'on veut générer, puis de relâcher la touche <ALT> ; aussitôt ce caractère apparaît à l'écran.

L'ASCII SUR LE QL

Dès sa conception le QL a été prévu pour être vendu à l'échelle mondiale ; par conséquent, les codes correspondant aux caractères graphiques ont été supprimés au profit de lettres internationales. C'est ainsi qu'une quantité impressionnante de caractères spéciaux sont disponibles, mais pas les caractères graphiques. Vous pouvez aussi les générer par l'intermédiaire du clavier. Par exemple <CTRL> <SHIFT>8 (les trois enfoncées en même temps) donne le caractère ">". Attention : ces codes ne sont pas nécessairement reconnus par votre imprimante.

L'ASCII ET VOTRE IMPRIMANTE

Un des gros intérêts de ce code est qu'on puisse s'en servir conjointement à une imprimante. A partir d'ARCHIVE, il faut d'abord envoyer un caractère ASCII nul (car(0)) puis la séquence de codes nécessaires au mode d'impression que vous désirez générer.

Exemple :

```
imprime
car(0)+car(27)+car(15)
```

pour spécifier à l'imprimante que l'on veut maintenant une impression en caractères élargis (sur l'imprimante EPSON FX 80).

imprime donne l'ordre d'envoyer les caractères à l'imprimante et non à l'écran (à moins que vous n'ayez modifié ceci par l'intermédiaire de la commande **via**,

car(0) est le caractère nul indispensable lorsque l'on s'adresse à l'imprimante,

car(27) est le code ASCII qui correspond à **<ESC>**, qui doit précéder tous les codes significatifs que l'on envoie à une imprimante EPSON,

+ est le signe de la concaténation ; nous envoyons des chaînes de caractères, pas des chiffres,

car(14) est le code qui signifie "écris maintenant en caractères élargis".

N.B. : ceci est un exemple pour l'imprimante

EPSON FX 80 ; relevez sur le manuel de votre imprimante les codes nécessaires à la sélection du mode d'impression que vous désirez.

car()
Syntaxe : car(exp. num.)
Fonction :

produit le caractère correspondant au code indiqué entre parenthèses. Certains caractères ne peuvent apparaître à l'écran car ils ne correspondent pas nécessairement à un code affichable (notamment les codes inférieurs à 32 qui sont réservés aux imprimantes). Au-delà du code 124, les caractères sont différents sur le QL et l'IBM PC.

Exemple : écris car(66)

affiche la lettre "B"

Emploi : cette fonction est utile pour faire apparaître les caractères spéciaux à l'écran et pour donner des ordres à une imprimante (voir encadrés).

Basic : CHR$()

code()
Syntaxe : code(exp. alpha.)
Fonction :

Cette fonction est complémentaire de **car()** ; elle produit l'effet inverse, à savoir donner le code ASCII du caractère figurant entre parenthèses. Si plusieurs caractères figurent entre parenthèses, seule la valeur du premier est renvoyée par **code()**.

Exemples : écris code("a")

donne : 97
que nom$="Bonjour"
écris code(nom$)

donne : 66 (valeur du code de "B")

Emploi : cette fonction est très employée pour avoir accès aux codes renvoyés par les touches de fonctions par exemple. Associée

à **clavier()** ou **touche()**, elle est précieuse ; on peut alors tester qu'une touche ou une combinaison de touches est actionnée.

Exemple : <CTRL> "A"

donne le code 1

Vous pourrez donc vous en servir dans vos applications en faisant des petits programmes du genre :

```
écris "POUR ARRÊTER TAPEZ SUR <CTRL> 'A'"
que test=code(clavier())
si test=1
stoppe
sinon
suiv
fsi
```

qui aura pour effet d'afficher les enregistrements successivement tant que vous ne taperez pas sur <CTRL> "A" en même temps.
code() est une fonction que vous devez absolument connaître.

Basic : CODE() / ASC()

* *les fonctions qui attendent des saisies au clavier*

Ces fonctions vous permettent, lors de l'exécution d'un programme, de l'interrompre momentanément pour vous permettre de rentrer une information au clavier. Vous pouvez par exemple avoir besoin d'automatiser la recherche d'un enregistrement et pour cela avoir besoin d'automatiser la recherche d'un enregistrement et pour cela vouloir que les messages s'affichent en clair à l'écran, que l'utilisateur n'ait plus qu'à rentrer le nom de la personne qu'il recherche et que la suite se fasse automatiquement.

clavier()
Syntaxe : clavier()
Fonction :

cette fonction arrête momentanément l'exécution d'un programme en attendant que vous tapiez **un** caractère au clavier. Le programme ne continuera pas son exécution tant que vous n'aurez rien tapé au clavier.

Exemple : que x$=clavier()

si vous tapez “a”, x$ sera affecté du caractère “a”.

Emploi : fonction également très utile, pour de multiples usages. En premier lieu, lorsque vous voulez par exemple que l'utilisateur entre une donnée par l'intermédiaire du clavier sans qu'il ait besoin de la valider par <RETURN>. Ceci n'est bien sûr valable que dans le cas où l'on ne doit rentrer qu'un seul caractère, puisque **clavier()** reprend l'exécution du programme juste après qu'un caractère a été tapé. En second lieu, cette fonction vous permet d'adapter le temps d'apparition d'une page de messages à l'écran à la vitesse de lecture de l'utilisateur qui devra taper sur une touche une fois qu'il aura tout lu et sera prêt à passer à la page suivante.

Exemple :

```
écris "LORSQUE VOUS AUREZ LU CECI APPUYEZ
SUR UNE TOUCHE"
que x$=clavier()
éponge
écris "VOUS AVEZ EU TOUT LE TEMPS NÉCES-
SAIRE POUR LIRE"
```

En outre, comme cela est expliqué dans les encadrés consacrés au code ASCII, **clavier()** vous permet de lire la pression d'une touche, ou d'une *combinaison de touches*.

Basic : INKEYS(−1) / GET$

touche()
Syntaxe : touche()
Fonction :

touche() produit le même effet que **clavier()**, à la différence qu'elle n'attend pas qu'un caractère soit tapé à l'écran. En d'autres termes, l'exécution d'un programme n'est pas suspendue par cette fonction.

Exemple : que x$=touche()

Ce qui a priori est la même chose qu'avec la fonction **clavier()**, mais x$ ne sera affecté d'aucune valeur si **au moment** où est rencontrée cette fonction aucune touche n'est enfoncée.

Emploi : cette fonction est surtout utilisée dans les boucles (voir

encadré). Il devient ainsi possible de tester le clavier et d'effectuer une autre tâche en même temps.

Exemple :

```
que x=0
tant que x<30
que x=x+1
que test$=touche()
si test$="a"
que x=31
sinon
écris "TAPEZ SUR 'A' POUR ARRÊTER"
fsi
ftantque
```

Cette petite procédure aura pour effet d'afficher en permanence "TAPEZ SUR 'a' POUR ARRÊTER" tant que l'utilisateur n'aura pas appuyé sur "a" ; de toute façon elle sera interrompue d'elle-même lorsque x aura atteint la valeur 30. C'est donc une boucle de temporisation que l'on peut arrêter à n'importe quel moment.

Basic : INKEY$(0) / INKEY$

LES BOUCLES

Les boucles sont très utilisées en informatique ; elles permettent de faire exécuter un nombre d'opérations plusieurs fois de manière répétitive. Ainsi par exemple l'expression **tantque/ftantque** permet de simuler une boucle.

Exemple :

```
que x=0
tantque x<100
que x=x+1
écris x
ftantque
```

Cette procédure aura pour effet d'afficher la valeur de x, qui *s'incrémente* à chaque passage de 1. Ainsi à l'écran apparaîtront toutes les valeurs de x entre 1 et 100 puisque nous avons spécifié que la boucle devait s'arrêter une fois que x aurait atteint la valeur 100.

Vous aurez remarqué que l'exécution de cette boucle prend un certain temps ; ici l'intérêt n'est pas évident, mais si nous supprimons **écris x** nous obtiendrons une *boucle de temporisation* qui interrompra l'exécution du programme **tantque**, x n'aura pas atteint la valeur 100. Ceci est parfois très utile

(pour laisser le temps à l'utilisateur de lire un message par exemple).

Il est aussi possible de réaliser des boucles qui ne s'interrompent pas :

```
que x=0
tanque x<1
écris « boucle interminable »
ftantque
```

Il n'y a aucune raison pour que cette boucle cesse puisque nous spécifions au départ que x est égal à 0 et qu'elle ne doit s'interrompre que lorsque x aura atteint la valeur 2. Or, étant donné que nous n'incrémentons pas x, cette boucle ne peut s'interrompre. Vous pouvez tout de même l'arrêter en tapant sur <**ESC**>, qui arrête arbitrairement toute exécution de programme.

* *les autres fonctions*

fdf() fdf(!nlf!)

Syntaxe :

Fonction :

cette fonction permet de tester si un fichier de données en est arrivé à la fin ou non. Lorsque le fichier a été lu jusqu'au bout, la valeur renvoyée par cette fonction est 1, dans le cas contraire c'est 0. Si plusieurs fichiers sont ouverts en même temps, il est possible de spécifier un *nom logique de fichier*.

Exemple : écris fdf()

Emploi : cette fonction est surtout utile dans une boucle. Ainsi on pourra exécuter un certain nombre d'opérations **tant que** le fichier n'aura pas été lu jusqu'au bout.

Exemple :

```
tantque fdf()<1
écris "IL RESTE DES ENREGISTREMENTS"
suiv
ftantque
```

qui aura pour effet de faire circuler les enregistrements un à un jusqu'à la fin du fichier.

En fait, à cette fonction peut être substituée une expression de la sorte :

```
que x=0
tantque x<nombenr()
que x=x+1
suiv
ftantque
```

qui produit le même effet, mais de façon plus lourde.

Basic : EOF()

sgn()
Syntaxe : sgn(exp.numé.)
Fonction :

indique le signe de l'expression numérique donnée entre parenthèses. Le résultat renvoyé par cette fonction est 0, si l'expression est nulle, 1 si elle est positive et −1 si elle est négative.

Exemple :

```
écris sgn(−60)  donne −1
écris sgn(0)    donne  0
écris sgn(100)  donne  1
```

Emploi : cette fonction n'est pas primordiale, d'autant qu'on peut très bien la remplacer facilement :

Exemple :

```
si x<0: écris "−1"
si x=0: écris "0"
si x>0: écris "+1"
```

Basic : SGN()

tache()
Syntaxe : tache()
Fonction :

donne le nom de la tâche courante dans laquelle on travaille. Cette fonction n'existe que sur l'IBM PC, puisque sur le QL il est impossible d'ouvrir plusieurs tâches simultanément.

Exemple : écris tâche()

qui affichera le nom que vous aurez donné à la tâche dans laquelle vous êtes.

Emploi : cette fonction est surtout utile lorsque l'on veut intégrer dans un programme une vérification de la tâche dans laquelle sont effectuées les opérations que l'on souhaite exécuter. En dehors de cela, **tâche()** ne présente pas un intérêt énorme, le nom de la tâche dans laquelle on travaille étant en permanence indiquée au bas de l'écran.

vu()
Syntaxe : vu()
Fonction :

permet de tester si une recherche est fructueuse ou non. Lorsque l'on recherche un enregistrement et qu'il en existe un qui correspond à l'expression que l'on recherche **vu()**, renvoie la valeur 1, dans le cas contraire **vu()**, renvoie la valeur 0.

Exemple : écris vu()

Emploi : cette fonction est surtout utile dans les boucles. En effet, pour automatiser une recherche, on a souvent besoin de tester si un enregistrement correspond au critère recherché ou non ; **vu()** permet de faire ceci.

2.7 LES MASQUES D'ÉCRAN

La plupart des logiciels permettant de gérer des bases de données imposent que l'on délimite la taille des champs et que l'on définisse un masque d'écran. ARCHIVE échappe à cette règle en ce sens que les champs n'ont pas de taille prédéfinie (si ce n'est 255 au maximum) et qu'un masque d'écran est déjà présent lorsque l'on crée un fichier.

LES MASQUES D'ÉCRAN

Un masque d'écran représente tout simplement la façon dont sont affichées les informations à l'écran. Lorsque l'on ne précise rien ARCHIVE utilise son propre masque d'écran où tous les champs sont alignés les uns en dessous des autres. Il est cependant possible de le modifier, de le sauvegarder pour une utilisation ultérieure.

2.7.1 Création d'un masque d'écran

Pour appeler le mode qui permet de créer un masque d'écran, il faut taper une nouvelle commande. Il s'agit de **éditem** (qui signifie

en abrégé *édite masque*). Lorsque vous validez cette commande, les différents champs apparaissent à l'écran (dans la zone d'affichage). Après quelques instants, une série de points apparaissent à leur tour à côté des champs. Ces points matérialisent *l'espace alloué* pour chacun d'entre eux. Dans le format standard (celui qu'ARCHIVE prend par défaut), l'espace alloué à chacun des champs est l'espace maximum que l'écran peut montrer — c'est-à-dire que les points vont jusqu'au bout de l'écran à droite.

A ce stade vous disposez de trois choix.

Entrer du texte, qui viendra s'afficher à l'écran et qui restera affiché quoiqu'il arrive.

Appuyer sur **<F3>** pour avoir accès au menu des commandes de l'éditeur d'écran.

Ou bien quitter le mode éditeur en appuyant sur **<ESC>** si vous voulez faire machine arrière.

* *entrer du texte*

Lorsque vous tapez du texte en étant dans le mode éditeur, celui-ci vient s'écrire à l'écran et fait partie intégrante du nouveau masque d'écran ; il restera toujours affiché à l'endroit que vous aurez défini.

* *F3*

Cette touche vous permet d'avoir accès aux commandes de l'éditeur d'écran ; cinq options vous sont proposées.

Abandon du masque (A)

Cette option vous permet de supprimer tout masque affiché à l'écran lorsque vous avez tapé **éditem**.

Choix d'encre (E)

Si vous disposez d'un moniteur couleur, cette commande vous permet de modifier la couleur du texte que vous allez taper à l'écran. Sur l'IBM PC, si vous ne disposez pas d'un moniteur couleur c'est un changement du style de l'écriture qui vous est proposé : normal, souligné, vidéo inverse, gras, gras et souligné. Sur le QL l'effet du changement de couleur sera de toute façon visible par des différences dans la brillance de certaines couleurs par rapport à d'autres.

Choix de papier (P)

C'est-à-dire changement du fond de l'écran. Attention, sur l'IBM PC ce choix ne vous est proposé que si vous avez installé votre système en précisant que vous disposiez d'un moniteur couleur (c'est l'option prise par défaut au moment de l'installation de XCHANGE). Sur le QL, les effets sont les mêmes que lors de la modification de l'encre, à savoir une différence dans la brillance de certaines couleurs.

Variables (V)

Cette option vous permet de préciser à quel endroit vous désirez voir s'afficher les informations contenues dans les champs. Dans ce cas vous devez positionner le prompt à l'endroit où vous désirez voir apparaître les informations, taper sur **<F3>**, taper sur **V**, rentrer le nom du champ qui contient les informations qui vous intéressent, puis valider avec **<RETURN>**. Ceci fait, ARCHIVE vous demande de taper sur n'importe quelle touche pour indiquer l'espace que vous désirez allouer à ces informations (choisissez toujours l'espace le plus large si rien ne vous en empêche) ; vous voyez alors apparaître des points qui symbolisent le nombre de caractères qui pourront venir s'afficher à cet emplacement. Pour terminer appuyer une nouvelle fois sur **<RETURN>**.

Cette opération n'est pas limitée aux emplacements des variables champs ; vous pouvez aussi affecter un emplacement pour une variable que vous avez vous-même déterminée, nous verrons comment faire un peu plus loin lors de l'exemple pratique.

* *suppression d'une variable*

Lorsque vous utilisez la présentation standard d'ARCHIVE tous les champs apparaissent à l'écran, mais vous pouvez par exemple décider, bien que votre fichier contienne une vingtaine de champs, de n'en afficher que dix. Pour cela vous devez en supprimer la moitié. Positionnez-vous à l'aide des flèches sur l'endroit où vous voulez supprimer une variable (ce n'est pas l'endroit où est écrit le nom du champ, mais l'emplacement des pointillés qui détermine où est située une variable champ), celle-ci apparaît en bas de l'écran dans la zone de contrôle ; tâpez sur **<F3>** et sur **V** ; si vous tapez immédiatement **<RETURN>**, le nom de la variable champ disparaît tout comme les points qui la matérialisaient ; vous pouvez maintenant la remplacer, entrer du texte, ou bien laisser cet emplacement libre.

* *sortie du masque d'écran*

Lorsque vous êtes satisfait de la nouvelle présentation que vous avez faite, quittez l'éditeur d'écran en appuyant sur **<ESC>**. Aussitôt, si vous avez un fichier ouvert et que les champs correspondent à ceux de ce dernier, les informations seront présentées de la façon dont vous venez de dessiner le nouvel écran.

Vous pouvez maintenant, s'il vous convient, le sauvegarder sur une disquette ou une cassette grâce à la commande **sauvem**. Cette commande vous propose de donner un nom au masque que vous venez de définir pour pouvoir le charger ultérieurement lorsque vous utiliserez le fichier pour lequel vous l'avez dessiné.

Il est fortement conseiller de sauvegarder les masques d'écran que l'on vient de créer, car celui-ci risque de disparaître à la suite d'une fausse manœuvre. Il faut garder à l'esprit qu'un masque d'écran défini par l'utilisateur vient se superposer à celui qu'ARCHIVE garde toujours en mémoire : le format standard. Ainsi, lorsque l'on tape **affiche** le format standard est activé et celui défini par l'utilisateur est éliminé purement et simplement. Par conséquent, il faut prendre soin de garder une version sur mémoire de masse (disquettes ou cassettes) afin de pouvoir le récupérer à tout instant sous peine de devoir tout redéfinir avec la commande **éditem**.

CONSEIL

Sauvegardez vos masques d'écran avec le même nom que celui du fichier pour lequel il a été conçu ; de toute façon lorsque vous le sauvegarder ARCHIVE ajoute automatique l'extension **SCN** qui le différencie du nom du fichier qui lui, porte l'extension **DBF**. Cela vous facilitera la tâche lorsque vous le rechercherez.

Tous les choix que nous vous avons expliqués plus haut sont accessibles à tout moment. En fait, lorsque vous désirez créer un masque d'écran, nous vous conseillons de procéder suivant une séquence bien précise :

Commencez par supprimer le masque d'écran initial en tapant :

<F3> A

Cette option a pour effet de vous laisser un écran vide et prêt à recevoir vos instructions.

Continuez en inscrivant le texte que vous voulez voir figurer à l'écran, ainsi que le cadre si vous souhaitez en mettre un.

Terminez par l'affectation des variables champs suivant l'espace disponible qui vous reste.

Sauvegardez votre travail.

Souvenez-vous que la commande **mode** vous permet de supprimer la séparation des différentes zones d'affichage et vous offre par conséquent une surface de travail beaucoup plus importante.

2.7.2 Les commandes associées aux masques d'écran

Nous avons vu qu'il était préférable de sauvegarder immédiatement le masque qui venait d'être créé.

Ceci peut être fait avec la commande **sauvem**.

```
sauvem          <RETURN>
"nom"           <RETURN>
```

Si l'on peut le sauvegarder, on peut bien sûr le recharger en mémoire ; il s'agit cette fois de la commande **chargem**.

```
chargem         <RETURN>
"nom"           <RETURN>
```

N.B. : il est inutile de préciser l'extension lors de ces opérations, ARCHIVE s'en charge automatiquement.

L'écran peut être « nettoyé », c'est-à-dire effacé comme un tableau noir, à l'aide de la commande **éponge**. Dans ce cas l'affichage du masque d'écran disparaît et vous devez le réactiver. Ne tapez surtout pas **affiche** comme nous pouvions le faire auparavant ; nous venons de voir que cette commande avait pour effet de supprimer le masque d'écran de la *mémoire* et revenait au mode d'affichage standard. Tapez en revanche **écran** qui active l'affichage que vous avez dessiné.

Enfin, lorsque vous voulez faire apparaître à l'écran des informations de manière automatique, vous devez utiliser l'instruction **montre** qui a pour but de *forcer* l'affichage. Ainsi pour voir apparaître les enregistrements les uns à la suite des autres, tapez :

```
partout:montre:fpartout
```

2.7.3 Application pratique

Reprenons le fichier que nous avons créé précédemment (celui que vous avez normalement appelé «carnet»), et créons un masque d'écran qui lui sera destiné.

Commençons par l'ouvrir, si ce n'est déjà fait :

```
ouvre "carnet"
```

puis activons l'affichage standard :

```
affiche
```

Ceci fait, entrons dans le mode d'édition de l'écran avec la commande **éditem**. Aussitôt le masque standard s'affiche à l'écran avec les points qui symbolisent l'espace réservé à chaque champ.

```
nom logique        : maître
nom$               :
prénom$            :
tel                : 0
```

Figure 2.5 : *Le masque d'écran standard*

Commençons, comme nous le recommandons, par supprimer totalement l'ancien écran :

```
<F3>    A
```

Nous allons en premier lieu encadrer l'écran avec le signe "*". Pour cela placez le prompt en haut à gauche et tapez une fois "*". Puis, en maintenant la touche majuscule enfoncée orientez-vous vers le bord opposé à celui où vous vous trouvez avec la flècle →. Arrivé au bout utilisez la touche ↓ pour descendre, et ainsi de suite jusqu'à la réalisation du cadre complet. N'oubliez pas de garder la touche majuscule enfoncée.

> En maintenant la touche <SHIFT> ou <MAJUS> enfoncée, et en actionnant les flèches de direction, le dernier caractère rentré au clavier est répété à l'infini.

Une fois cette opération réalisée, nous allons mettre un titre en

haut et au centre. Positionnez donc le prompt en haut et un peu sur la gauche et tapez en majuscules :

CARNET DE NOMS ET DE NUMÉROS DE TÉLÉPHONES

descendez de quelques lignes et entrez le libellé du premier champ :

PRÉNOM:

puis quelques espaces plus loin, tapez le second libellé :

NOM:

Remarquez que nous n'avons pas porté d'indication quant à la nature des champs nom$ et prénom$. Pour l'instant nous nous contentons de taper le texte qui servira à localiser les informations qui apparaîtront et ce texte ne doit pas être confondu avec les variables champs qui, elles, devront porter le signe "$" exactement comme cela figure dans le nom du champ. Nous nous occuperons de cela dans un instant.

Il reste encore un champ, que nous allons placer deux lignes en dessous du mot NOM:

TÉLÉPHONE:

Enfin, comme nous aimons la difficulté, nous allons ajouter une variable qui n'a aucun rapport avec le fichier (plaçons le texte encore deux lignes en-dessous du mot TÉLÉPHONE :

HEURE:

Il ne reste plus maintenant qu'à indiquer l'emplacement des variables et l'espace qui leur est alloué.

Positionnez le prompt après le mot PRÉNOM, laissez un espace et tapez sur <**F3**>. Tapez ensuite sur **V** et entrez le nom de la variable champ qui contient les prénoms :

prénom$

Remarquez que ce dernier vient s'inscrire en bas de l'écran ; validez-le et lorsqu'ARCHIVE vous y invite, appuyer sur n'importe quelle touche (excepté <**RETURN**>) pour délimiter le nombre de caractères que vous allouez à ce champ ; vous pouvez laisser une

dizaine de points puisque vous êtes rapidement limité par le mot NOM, tapez sur <**RETURN**> pour valider l'opération et placez le prompt à droite du mot NOM et recommencez l'opération que nous venons de faire, avec cette fois-ci la variable champ nom$. Et procédez de manière identique pour le champ tel, deux lignes plus bas. Enfin pour la variable correspondant au mot HEURE, nous allons utiliser la fonction décrite dans la section précédente et qui donne l'heure courante (peu importe que vous ayez mis l'horloge interne à l'heure ou pas, ceci n'est qu'un exemple).

Lorsque vous vous serez positionné après le mot HEURE (toujours en laissant un espace), tapez sur <**F3**> et sur **V**, puis introduisez le nom de la fonction.

```
heure()
```

Ceci fait procédez de manière identique ; validez la variable, tapez sur une touche pour délimiter la place allouée et tapez à nouveau sur <**RETURN**>.

Quittez enfin l'éditeur d'écran et sauvegardez immédiatement votre masque avec la commande **sauvem**.

Maintenant votre masque d'écran est totalement achevé et prêt à être utilisé aussi bien pour afficher les enregistrements que pour en rentrer de nouveaux ou les modifier. Avouez que cette présentation est tout de même plus fonctionnelle que celle que nous avons vue jusqu'ici.

Nous avons vu jusqu'ici la majorité des commandes et fonctions d'ARCHIVE. Pour des applications tout à fait classiques de gestion de fichiers, ce que nous avons vu est parfaitement suffisant. Cependant, ARCHIVE offre de nombreuses autres possibilités lorsque l'on sait programmer. ARCHIVE permet en effet d'automatiser au maximum les opérations, vous rendant ainsi la vie encore plus facile. Les chapitres suivants s'adressent plutôt aux personnes qui ont des rudiments de programmation, mais si vous avez pu suivre toutes les opérations que nous avons effectuées jusqu'ici sans problème, il n'y a aucune raison que vous ne goûtiez pas aux commodités des programmes.

N'oubliez pas : lorsque vous terminez une session, n'éteignez pas votre appareil sans avoir au préalable refermé votre fichier avec la commande **ferme**.

CHAPITRE 3

Programmation

3.1 POURQUOI LA NÉCESSITÉ DE PROGRAMMER ?

A priori, on voit difficilement comment il est possible d'améliorer encore les performances d'ARCHIVE. Et cependant, la programmation et l'utilisation simultanée de plusieurs fichiers vont nous être d'utilité, surtout lorsque nous allons aborder les applications pratiques dans les deux derniers chapitres.

Le premier rôle des programmes écrits sous ARCHIVE est de permettre une plus grande automatisation des opérations que vous réalisez souvent ; les recherches et les tris que nous avons faits jusqu'ici n'ont pas été pénibles car ils portaient sur un fichier de très petite taille, mais imaginez les mêmes opérations sur des fichiers de plusieurs milliers d'enregistrements... Il est sans aucun doute plus commode d'automatiser certaines tâches le plus possible afin de gagner encore plus de temps à l'exécution.

En second lieu, il se peut que les personnes qui auront à se servir des données que vous aurez rentrées ne soient pas aussi averties que vous du fonctionnement d'ARCHIVE. Editer des étiquettes, par exemple, est un problème courant et facile à réaliser, mais comment l'obtenir de personnes non familiarisées avec ARCHIVE ? La réalisation d'un petit programme réalisant ceci automatiquement va permettre à n'importe qui de le faire à votre place.

Enfin, dernier agrément de la programmation, ARCHIVE va vous permettre de créer des applications « sur mesure ». Aucun besoin de logiciels supplémentaires, lorsque celui que vous avez sous les yeux va vous donner la possibilité de résoudre une part importante des problèmes qui se posent à votre société, votre cabinet, ou bien encore votre société d'études (nous donnons à la fin de ce guide un exemple d'une application qui permet d'analyser les résultats d'une enquête en un temps record).

Et puis, détail qui a de l'importance, les programmes que vous réaliserez sous ARCHIVE offriront plus de convivialité à l'utilisateur et nul ne saurait s'en plaindre.

Si vous avez déjà programmé dans un langage quelconque (le Basic par exemple), vous n'aurez aucun mal à comprendre la façon d'utiliser les instructions d'ARCHIVE. Les personnes n'ayant aucune notion de programmation ne doivent cependant pas abandonner en si bon chemin, il suffira d'un peu plus de temps, mais les résultats seront les mêmes.

3.2 LES INSTRUCTIONS DU LANGAGE DE PROGRAMMATION D'ARCHIVE

Le langage de programmation d'ARCHIVE est de conception modulaire et procédurale. Cela signifie que pour fonctionner correctement les programmes doivent être *structurés* et fragmentés en différentes tâches. Chacune d'entre elles pourra en appeler une autre et ainsi de suite. Ces tâches sont appelées procédures, et sont reconnaissables par leur nom. En outre les procédures d'ARCHIVE ont la possibilité de s'appeler elles-mêmes (récursivité). Chaque procédure que l'on crée est comme une nouvelle commande ou instruction.

Nous avons vu dans les chapitres précédents un certain nombre de fonctions dont dispose ARCHIVE, en voici quelques autres qui vont nous permettre de programmer. Nous donnons ci-dessous leur nom, leur syntaxe, un exemple de leur utilisation, ainsi que leur correspondant en Basic quand cela est possible.

N.B. : désormais, nous ne parlerons plus de commandes, comme dans les chapitres précédents, mais d'instructions.

En théorie, la plupart des mots que l'on peut visualiser dans la zone de contrôle grâce à la touche **<F3>** sont aussi des instructions. En fait certaines d'entre elles ne peuvent être utilisées dans un programme, car elles rompent l'exécution de celui-ci (ex. : modifie).

écris
Syntaxe : écris <exp.>
Fonction :

affiche à l'écran l'expression qui suit. Un point-virgule placé à la suite de l'expression à imprimer empêche un saut de ligne. Il n'y a pas de tabulation automatique avec "," ; **écris** dispose de plusieurs auxiliaires, qui permettent de positionner l'affichage sur l'écran :

au qui permet de positionner l'affichage selon des coordonnées exprimant une intersection de lignes et de colonnes.

écris au 0,0; "Bonjour"

a pour effet d'afficher "Bonjour" dans le coin gauche en haut de l'écran.

tab qui déplace sur *la droite* le curseur selon le nombre de caractères spécifiés après tab.

tab(10): écris "tab10"

remplit les 10 premiers caractères par un espace (et par conséquent efface tout ce qui y figurait) et affiche "tab 10".

encre qui modifie la couleur ou l'attribut de l'affichage du message après **écris**. Voir **encre** pour la syntaxe exacte.

encre 2: écris "encre rouge"

affiche le message « encre rouge » en rouge ou en gras pour le monochrome.

papier qui modifie le fond sur lequel sera affiché l'expression.

papier 4: écris "fond vert"

affiche le message "fond vert" sur un fond vert ou reste sans effet sur un écran monochrome.

Basic : PRINT (AT, LOCATE, TAB, INK, PAPER, STRIP)

imprime
Syntaxe : imprime <exp>
Fonction :

fonctionne exactement de la même façon qu'**écris** ; toutefois l'expression à imprimer n'est pas affichée à l'écran mais dirigée vers l'imprimante. Les auxiliaires d'affichage décrits dans **écris** sont exactement les mêmes sauf pour ceux concernant la couleur.

Exemple : imprime "Message pour l'imprimante"

Basic : LPRINT

encre
Syntaxe : encre <exp. num.>
Fonction :

modifie jusqu'à nouvel ordre la couleur ou l'attribut de l'affichage à l'écran. L'expression numérique après **encre** correspond aux couleurs ou attributs suivants :

0	noir
1	bleu
2	rouge
3	violet
4	vert
5	bleu clair
6	jaune
7	blanc

Sur l'IMB PC, en monochrome

0	Souligné
1	vidéo inverse
2	gras
3	gras et souligné
4	normal

Exemple : encre 7

Basic : INK

papier
Syntaxe : papier <exp. num.>
Fonction :

permet de changer, jusqu'à nouvel ordre, la couleur du fond. Les couleurs correspondent à la valeur que prend l'expression numérique après **papier**. Voir l'instruction **écrit** pour le code des couleurs. Si vous ne disposez pas d'un moniteur couleur sur l'IBM PC, cette instruction est sans effet.

Exemple : papier 5

Basic : PAPER

éponge
Syntaxe : éponge
Fonction :

permet de nettoyer l'écran. L'affichage est alors désactivé et il faut utiliser **affiche** ou **écran** pour réactiver l'écran.

Exemple : éponge

Basic : CLS, HOME

erreur
Syntaxe : erreur <proc>
Fonction :

permet de se diriger vers une procédure de traitement, lorsqu'une erreur dans l'exécution d'une procédure se produit. Une routine spéciale peut être appelée selon chaque type d'erreur ; **numerr()** permet de repérer quelle est la nature de l'erreur selon un nombre et par conséquent de les traiter séparément.

Exemple : erreur proc1

on peut alors concevoir la procédure proc1 ainsi :

```
proc1
  si numerr()=1
    écris « VOUS AVEZ FAIT UNE ERREUR DE SYNTAXE »
```

```
        sinon
        écris « VOUS AVEZ FAIT UNE AUTRE ERREUR »
      fsi
    fproc
```

Basic : RESUME

exéc
Syntaxe : exéc !objet! <nfp>
Fonction :

charge en mémoire vive le programme spécifié et exécute automatiquement la procédure nommée "start". Vous pouvez ajouter le nom facultatif "objet" et le programme sera chargé en binaire (beaucoup plus rapidement) plutôt qu'en ASCII. Il aura dû être sauvegardé dans ce mode.

Exemple : exéc objet "mesproc"

Basic : sur l'IBM PC cela correspond au démarrage automatique grâce au fichier AUTOEXEC.BAT. Sur le QL au programme Boot.

Les programmes peuvent être sauvegardés en ASCII ou en binaire. Dans le second cas, le transfert est beaucoup plus rapide étant donné qu'aucune conversion n'est nécessaire. Il faut, lorsqu'on le sauvegarde, préciser à l'aide du mot **objet** la nature du transfert.

liste
Syntaxe : liste
Fonction :

permet de lister toutes les procédures en mémoire sur une imprimante, ou bien sur un autre périphérique si cela est précisé par la commande **via**.

Exemple : liste

Basic : LIST

N.B. : il n'est pas possible de lister certaines procédures sélectionnées, toutes celles qui sont en mémoire vive sont listées.

local
Syntaxe : local <var>, <var>...
Fonction :

sert à déclarer une variable locale à une procédure. Cela signifie que seule la procédure où elle est déclarée la reconnaîtra. Les variables qui ne sont pas déclarées *locales* sont appelées *globales* et sont communes à toutes les procédures. Les variables locales doivent être déclarées comme telles à la première ligne de la procédure impérativement.

Exemple : local x, y, z

Basic : LOCAL

normal
Syntaxe : normal
Fonction :

Permet de rediriger les commandes **imprime**, **état**, **liste** vers l'imprimante. Ceci peut s'avérer nécessaire si vous avez modifié le canal de sortie avec la commande **via**.

Exemple : normal

Basic :

via
Syntaxe : via <nfp> !exporte! état!
via écran
Fonction :

cette instruction permet de modifier le canal d'affichage des instructions **imprime**, **état** ou **liste**. Les données peuvent être écrites sur un fichier ou bien apparaître à l'écran selon l'expression qui suit **via**. Si vous dirigez les données vers un fichier, celles-ci passent tout de même par le gestionnaire d'imprimante et contiennent par conséquent tous les codes nécessaires à cette dernière. Si vous précisez **exporte**, les données sont alors « filtrées » et il ne reste plus que les codes de retour chariot et de saut de ligne, ce qui permet de les importer facilement dans une autre application.

Enfin, si vous précisez **état**, les données ne passent pas du tout

par le gestionnaire d'imprimante et ne sont pas filtrées ; elles contiennent par conséquent leurs codes spécifiques.

Vous pouvez aussi spécifier **écran** après via, ce qui a pour effet de diriger toutes les données vers l'écran.

Exemple : via écran

Basic : —

si
sinon
fsi
Syntaxe : si <exp>:... !sinon: ...! fsi
Fonction :

exécute une ou plusieurs instructions **si** une condition est vérifiée, **sinon** exécute une autre série d'instructions. Le **sinon** est facultatif, mais **fsi** doit obligatoirement terminer ce genre de test, sinon une erreur (de structure) serait générée.

Exemple : si a=1: écris "a=1": sinon: écris "a<>1": fsi

Basic : IF THEN ELSE

trace
Syntaxe : trace
Fonction :

permet de suivre l'exécution d'un programme pas à pas ; chaque ligne est affichée à l'écran lors de l'exécution de celui-ci. Lorsque **trace** est rencontré pour la première fois le mode pas à pas est activé, lorsqu'il est rencontré pour la deuxième fois, il est désactivé. Si vous voulez suspendre momentanément l'exécution d'un programme, il suffit de maintenir la barre d'espace enfoncée et de la relâcher pour repartir.

Lorsque l'on exécute un programme, le fait de presser sur **<F3>** active et désactive le mode pas à pas.

Exemple : trace

Basic : TRON, TROFF

que
Syntaxe : que <var> = <exp>
Fonction :

nous l'avons déjà utilisée ; **que** est l'instruction d'affectation et sert à initialiser une variable avec l'expression qui suit le signe "=". Contrairement à son emploi dans certains langages *elle n'est pas facultative.*

Exemple : que x=5

Basic : LET

note
Syntaxe : note <message>
Fonction :

permet de mettre des commentaires dans un programme. Ce commentaire ne sera pas pris en compte par le programme pendant l'exécution.

Exemple : note Ceci est un commentaire

Basic : REM

stoppe
Syntaxe : stoppe
Fonction :

Arrête l'exécution d'un programme et repasse en mode direct. **continue** ne peut être utilisé ici pour relancer l'exécution.

Exemple : si test=0: stoppe fsi

Basic : STOP (BREAK)

tantque/ftantque
Syntaxe : tantque : <exp>...: ...: ftantque
Fonction :

permet d'exécuter une ou plusieurs instructions tant qu'une condition est vraie.

Exemple :

```
tanque i<30
que i=i+1
écris i
ftantque
```

Basic : WHILE, END WHILE

partout/fpartout
Syntaxe : partout !<nfl!: ... : fpartout
Fonction :

permet de réaliser toutes sortes d'opérations sur *l'ensemble d'un fichier*, c'est une instruction que nous avons déjà vue plusieurs fois et qui devrait vous être familière.

Exemple : partout: montre: fpartout

Basic : -

Vous aurez sans doute noté la présence des deux points ":" pour séparer deux instructions entre elles. Cette ponctuation est indispensable et une erreur de syntaxe serait générée s'ils manquaient.

LES ERREURS DE SYNTAXE...

La syntaxe d'une ligne est une chose extrêmement rigoureuse dans ARCHIVE. Si vous tapez une lettre pour une autre, si vous omettez deux points, ou si vous tapez un mot qu'ARCHIVE ne connaît pas un message d'erreur est immédiatement indiqué. Dans ce cas appuyez sur **<F5>** pour récupérer la ligne qui a produit l'erreur et corrigez-la.

Nous venons de voir les instructions qui servent à programmer ARCHIVE et il serait souhaitable de pouvoir les exploiter dès à présent dans de petits programmes. Il est auparavant indispensable de connaître la façon dont on peut créer ces procédures. Pour cela ARCHIVE dispose d'un mode supplémentaire (éditeur de programme) que nous allons à présent examiner dans les moindres détails.

3.3 COMMENT PROGRAMMER ARCHIVE

Lorsque vous créez un programme pour la première fois, il suffit de taper **édite** (ne confondez pas avec **éditem** qui est l'éditeur de masque d'écran) pour entrer dans le mode *édition de procédures*.

Aussitôt le prompt vous invite à rentrer le nom de la procédure que vous voulez créer. Nous allons choisir le nom arbitraire "b".

Le but de la procédure sera d'afficher à l'écran le message :

voici ma première procédure

tapez b, puis validez avec la touche <**RETURN**> immédiatement l'éditeur vous invite à rentrer la première ligne qui va composer votre procédure. Notez la présentation. Cette ligne va venir s'inscrire entre **proc b** et **fproc**. Mais dans un premier temps (c'est-à-dire tant que vous ne l'avez pas validée avec <**RETURN**>) les caractères que vous taperez s'inscriront dans le bas de l'écran ; tapez donc la ligne nécessaire à l'opération que nous voulons voir exécuter :

écris « voici ma première procédure »

puis validez-la avec <**RETURN**>. Vous l'avez constaté, la ligne est venue se « positionner » entre **proc b** et **fproc**, le prompt repasse à nouveau au bas de l'écran, vous invitant à taper une seconde ligne qui viendrait — après validation — s'insérer en dessous de celle que nous venons de rentrer. Mais notre ligne suffit et nous pouvons quitter le mode *édition de procédures*, pour revenir au mode direct. Pour cela, il faut taper sur <**ESC**> deux fois de suite (une fois pour sortir du mode insertion de lignes et une autre fois pour quitter l'éditeur de procédures).

A ce stade, vous venez sans le savoir d'ajouter une instruction de nom "b" et qui a pour effet d'afficher "voici ma première procédure" à l'écran. Rien d'extraordinaire pour le moment, mais nous allons tout de même l'exécuter.

Tapez : b <RETURN>

et le message apparaît à l'écran !

Recommencez : b <RETURN>

et le message est imprimé une seconde fois.

Vous pouvez recommencer autant de fois que vous le désirez, cela fonctionnera à tous les coups ; vous avez bel et bien créé une nouvelle instruction.

Pour ne pas risquer de perdre un travail aussi remarquable (!), nous allons tout de suite sauvegarder notre procédure sur disquette ou cassette :

L'instruction qui permet de faire ceci est **sauve**

Syntaxe : sauve !objet! protégé! <nfp>
Fonction :

sauve vous permet de sauvegarder les procédures résidants en mémoire, vous devez donner un nom physique (qui peut être le même que celui d'un fichier puisque l'extension est différente). En outre **sauve** vous permet de sauvegarder des programmes **objets** ou en ASCII, mais les programmes objets sont plus rapides à sauvegarder. Vous pouvez aussi spécifier que vous voulez le sauvegarder de façon qu'on ne puisse le lister ni le modifier ; dans ce cas, il faut préciser **protégé** avant le nom du fichier (uniquement sur IBM PC).

Exemple : sauve "maproc"

et notre procédure sera sauvegardée sous le nom "maproc" avec le format ASCII et en version non protégée.

Pour la recharger en mémoire nous devrons utiliser **charge**

Syntaxe : charge !objet! <nfp>
Fonction :

recharge en mémoire vive un programme déjà sauvegardé sur la mémoire de masse. Si votre programme a été sauvegardé avec la mention **!objet!**, vous pouvez aussi le recharger de cette manière en le précisant.

Exemple : charge objet maproc

charge le programme "maproc" en binaire (il aura dû être sauvegardé de cette façon).

LES EXTENSIONS

Lorsque ARCHIVE sauvegarde un programme ou bien crée un fichier, les noms que vous leur donnez (ex. : "maproc", "carnet", etc.) se voient ajouter des *extensions*. Ces extensions sont constituées de trois lettres qui sont alors « collées » au nom que vous donnez au fichier à sauvegarder avec un *séparateur* (un tiret sur le QL "-" et un point sur l'IBM PC "."). Pour vérifier cela demandez le « catalogue » de votre disquette ou cassette avec la commande **cat** suivie de deux fois <**RETURN**>.

Ex. : Maproc_prg

qui signifie « programme ».

Ex. : carnet.dbf

qui signifie « fichier ».

ARCHIVE ajoute une *extension* pour tous les fichiers et programmes que vous transferez sur la mémoire de masse. Cela permet au logiciel de retrouver facilement où placer le fichier qu'on lui demande de lire. Ces extensions ne sont pas indispensables à connaître, mais elles vous permettent à vous aussi de retrouver à quoi correspondent les fichiers qui figurent sur votre mémoire de masse.

Vous savez désormais comment créer des procédures (aussi simples soient-elles) mais le fait de pouvoir afficher « voici ma première procédure » plusieurs fois à l'écran ne présente pas un avantage notable.

Nous allons donc, pour pratiquer cette nouvelle découverte, vous donner des exemples de procédures plus complexes, mais aussi plus puissantes par leur fonction.

3.4 DES PROCÉDURES PLUS SOPHISTIQUÉES

3.4.1 Simplifier des commandes

Vous le remarquerez à l'usage, lorsque vous manipulez des fichiers, le fait de taper des commandes en entier prend beaucoup de temps, surtout lorsqu'il s'agit de répéter plusieurs fois les mêmes commandes. Rappelez-vous qu'il est toujours possible de rappeler la dernière en appuyant sur <**F5**> et en validant de nouveau avec <RETURN>, mais attention, cela n'est valable que si vous ne tapez rien entretemps. Pour y remédier voici des procédures qui vous permettent de simplifier des commandes à la frappe d'une lettre seulement.

Après la constitution d'un fichier, vous souhaitez revoir chaque enregistrement successivement pour vous assurer qu'il ne susbiste aucune faute (il y en a toujours !). Pour cela vous avez besoin d'utiliser souvent les commandes suivantes :

suiv	pour faire défiler les enregistrements,
préc	pour revenir sur un enregistrement que vous avez passé trop rapidement

cherche	pour rechercher un enregistrement particulier,
modifie	pour rentrer dans le mode modifie et faire les corrections.

Chacune de ces commandes commence par une lettre différente, nous allons donc appeler nos procédures par les initiales de celles-ci.

suiv

Commencez par rentrer dans le mode éditeur de procédures :

édite

Aussitôt, étant donné qu'il n'y a pas de procédure en mémoire, vous rentrez dans le mode *création de procédures*. C'est-à-dire que la ligne du bas attend que vous donniez un nom. Tapez donc le nom de la première procédure :

s

Puis validez avec <RETURN>. Vous pouvez maintenant commencer à rentrer la première ligne (qui, rappelons-le, s'inscrit d'abord en bas de l'écran).

suiv

Tapez maintenant sur <**ESC**> pour sortir du mode *création de procédures*, le guide (zone supérieure) vous montre maintenant les possibilités que vous avez :

appuyer sur	<F3>
appuyer sur	<F4>
appuyer sur	<ESC>

La première opération a pour objet de vous donner accès aux options de l'éditeur de procédures — nous y reviendrons dans la partie suivante.

La deuxième vous permet de repasser dans le mode insertion de lignes, si vous désirez ajouter des lignes en supplément de celle que vous avez écrite (suiv) ; il suffit à l'aide des flèches de vous positionner juste au-dessus de l'endroit où vous voulez insérer la ligne et d'appuyer sur <**F4**>. Vous repassez à ce moment dans le

mode insertion et vous y restez tant que vous ne tapez pas sur <**ESC**>.

La troisième solution (<F5>) vous permet de revenir sur une ligne que vous avez déjà écrite et que vous désirez corriger. Si vous désirez effectuer une correction, positionnez-vous à l'aide des flèches sur la ligne que vous désirez modifier et taper sur <**F5**>. Aussitôt la ligne réapparaît en bas de l'écran et vous avez la possibilité de la corriger grâce aux commandes de l'éditeur décrite dans la section 2.2. Une fois votre correction effectuée, validez à nouveau la ligne avec <**RETURN**> et celle-ci disparaît du bas de l'écran pour réapparaître à nouveau dans la partie supérieure avec la correction que vous y avez apportée.

La quatrième option (<ESC>) sert à sortir du mode *éditeur de procédures*, et à retourner au mode direct.

Vous pouvez dès maintenant essayer les effets de votre procédure. Ouvrez un fichier (par exemple celui que nous avons créé dans le second chapitre), et tapez :

```
affiche <RETURN>
```

Le premier enregistrement apparaît alors. Pour accéder au suivant tapez :

```
s     <RETURN>
```

et aussitôt l'enregistrement suivant est affiché, grâce à la procédure que vous venez de créer. Vous avez ajouté un mot qu'ARCHIVE reconnaît ! Cela ne signifie pas que **suiv** n'est plus valable ; vous pouvez toujours avancer dans le fichier avec cette commande, cela produira le même effet, mais il faudra que vous tapiez toutes les lettres, ce qui est tout de même plus long que **s**.

Mettons tout de suite en pratique les options que l'on nous proposait à l'étape précédente ; tapez :

```
édite <RETURN>
```

Vous êtes à nouveau dans *l'éditeur de procédures* mais il y a déjà une procédure, vous ne passez pas directement dans le mode *création de procédure* comme lorsque nous sommes entrés dans ce mode la première fois.

Nous allons la modifier de façon qu'elle indique clairement à l'écran qu'il n'y a plus d'enregistrement dans le fichier une fois que nous aurons atteint le dernier à force de faire plusieurs fois **s**.

A l'aide des flèches, positionnez-vous sur la ligne où est écrit

proc s ; nous allons insérer une nouvelle ligne juste au-dessous (juste avant **suiv**). Tapez sur <**F4**>, qui permet d'insérer de nouvelles lignes. Tapez maintenant la nouvelle ligne :

si fdf()<1 <RETURN>

puis ressortez du mode *insertion de ligne* en tapant une fois sur <**ESC**>, descendez à nouveau, à l'aide des flèches, et positionnez-vous sur la ligne **suiv** ; nous allons maintenant insérer une nouvelle ligne en dessous de cette dernière. Tapez donc à nouveau sur <**F4**> et tapez la nouvelle ligne.

sinon <RETURN>

et la suivante :

écris au 10,10 ; "DERNIER ENREGISTREMENT" <RETURN>

et encore celle-ci :

fsi <RETURN>

Rappelez-vous : une condition **si** doit toujours être terminée par **fsi**. Tapez maintenant sur <**ESC**> deux fois de suite (une fois pour revenir au mode édition de procédures et une autre pour revenir au mode direct). Essayez à présent de vous en servir (tapez **affiche** pour faire réapparaître le fichier) ; a priori rien de différent par rapport à ce qui se passait avant, sauf que lorsque vous arrivez au dernier enregistrement, le message que nous avons écrit dans la procédure apparaît à l'écran.

Nous allons maintenant créer la seconde procédure :

Tapez **édite** une nouvelle fois, vous voilà de nouveau dans le mode *édition de procédure*, il faut maintenant indiquer à ARCHIVE que l'on veut créer une nouvelle procédure ; tapez sur <**F3**>. Aussitôt les options disponibles apparaissent :

A — pour **A**bandonner une procédure
E — pour **E**xtraire une ligne
I — pour **I**njecter une ligne
C — pour **C**réer une nouvelle procédure

Le premier choix, **A**, vous permet de supprimer une procédure de la mémoire. Attention, à moins que vous ne l'ayez sauvegardée sur la mémoire de masse auparavant, il ne sera pas possible de la

récupérer (si ce n'est en la retapant entièrement). Pour supprimer une procédure, il faut tout d'abord indiquer laquelle (s'il y a plusieurs procédures en mémoire) ; vous devez faire ceci *avant* d'appuyer sur <**F4**>. Pour cela vous disposez des touches <**TAB**> et <**SHIFT**> <**TAB**> (<**MAJUS**> sur les appareils français), qui vont vous permettre de sélectionner les procédures. En appuyant sur <**TAB**> les procédures sont affichées successivement de la première vers la dernière. En appuyant sur <**SHIFT**> <**TAB**>, les procédures sont listées de la dernière vers la première. Choisissez donc la procédure que vous souhaitez supprimer et appuyez sur <**F3**>, puis sur **A** et tapez sur <**RETURN**> pour confirmer que vous souhaitez bien supprimer cette procédure, ou bien sur une autre touche si vous vous ravisez. Immédiatement, la procédure disparaît définitivement.

Le choix, **E**, vous propose d'extraire une ligne et de la garder en mémoire jusqu'à ce qu'une nouvelle ligne soit extraite et à son tour gardée en mémoire. La manipulation est simple : avant de taper sur <**F3**>, choisissez la ligne que vous souhaitez extraire à l'aide des flèches. Ceci fait, tapez sur <**F3**> puis sur **E**, actionnez à nouveau les flèches si vous désirez extraire plusieurs lignes, puis tapez sur <**RETURN**>, la ou les lignes disparaissent et se trouve maintenant en mémoire : **elle y restera tant que vous n'extrairez pas de nouvelles lignes**. Vous pouvez la réinsérer à un autre endroit dans la procédure, ou bien même dans une autre procédure grâce à l'autre option qui vous est proposée lorsque vous appuyez sur <**F3**>.

Cette troisième option, **I**, autorise donc la réinjection de la ligne que vous venez d'extraire et qui se trouve momentanément en mémoire. Pour la réinjecter, choisissez la procédure dans laquelle vous voulez le faire (toujours avec <**TAB**> et <**SHIFT**> <**TAB**>) puis placez-vous à l'aide des flèches au-dessus de la ligne où vous voulez la réinjecter. Appuyer sur <**F3**> puis sur **I** et la ligne est réinjectée. La mémoire est vide.

C, la quatrième option, est celle qui vous permet de créer une nouvelle procédure. Tapez sur <**F3**> puis sur **C**, vous constaterez immédiatement que la procédure qui était affichée disparaît et que le prompt vous invite à rentrer le nom d'une nouvelle procédure. Ceci fait, son nom vient s'inscrire à côté de celle(s) qui existe(nt) déjà et vous vous retrouvez immédiatement en mode *insertion de lignes* comme si vous aviez tapé sur <**F4**>. Insérez les lignes que vous désirez insérer puis tapez sur <**ESC**> pour en ressortir et vous retrouver au mode *éditeur de procédures*.

Par conséquent, pour créer la nouvelle procédure que nous souhaitions créer, il vous suffit de taper sur <**F3**> puis sur **C**.

Donnez le nom de cette procédure :

```
P
```

puis écrivez la ligne :

```
préc
```

sans oublier de valider à chaque fois.

Sortez du mode insertion puis du mode édition (deux fois <ESC>) et testez votre nouvelle procédure :

```
P <RETURN>
```

Les enregistrements défilent bien de la fin vers le début.

La troisième procédure fonctionne sur le même principe ; vous devriez être capable de la créer vous-même. Rappelons cependant brièvement l'ordre des opérations.

```
édite
<F3>
C
Nom de la procédure (m)
insertion de la ligne (modifie)
<ESC>
<ESC>
```

Essayez à nouveau cette nouvelle procédure :

```
m <RETURN>
```

et vous voyez apparaître le prompt en haut de l'écran, à côté du premier champ, comme expliqué dans le chapitre 2.

La dernière procédure est un peu plus complexe et nous allons l'expliquer plus en détail. En premier lieu, créez la procédure suivante selon les opérations maintenant habituelles :

```
proc c
saisis au 10,0; "QUEL NOM CHERCHEZ-VOUS ?" ; n$
cherche nom$=n$
si vu()
affiche
sinon
écris au 10,0; "CE NOM N'EXISTE PAS"
```

```
fsi
fproc
```

Lorsque vous utiliserez cette procédure, que va-t-il se passer ? En premier lieu, le prompt vous invitera à rentrer le nom que vous cherchez. Remarquez que nous avons utilisé l'instruction **saisis** et que nous lui avons demandé de poser la question à un endroit précis de l'écran (les coordonnées **10,0**) que vous pouvez bien sûr modifier. Lorsque vous tapez ce nom et que *vous le validez* avec <RETURN>, la variable alphanumérique **n$**, est affectée du nom que vous avez tapé. L'instruction suivante est la commande classique pour rechercher un nom dans un fichier et **vu()** sert à tester si effectivement un nom tel que celui que vous cherchez existe bien dans un enregistrement, **affiche** sert à forcer l'affichage au cas où celui-ci serait désactivé. Dans le cas où aucun nom ne correspond (**sinon**), le message « CE NOM N'EXISTE PAS » est affiché au même endroit que la question précédente.

Voilà, nous venons de créer la première procédure vraiment utilisable par n'importe qui — il n'est pas besoin de connaître les commandes **cherche** ou **trouve** pour retrouver un nom et c'est tout de même plus pratiques pour les personnes ne connaissant pas le maniement d'ARCHIVE.

3.4.2 Automatisation des tâches

Nous avons vu au cours des précédents chapitres comme il était facile de faire exécuter une ou plusieurs tâches de manière répétitive sur l'ensemble d'un fichier. Nous donnons ici la procédure qui permet de transformer les anciens numéros de téléphone (à sept chiffres) en nouveaux à huit chiffres.

Nous allons tout d'abord supposer que vous disposez déjà d'un fichier dont les champs sont organisés comme ci-dessous :

```
nom$     : Durand
prénom$  : Jean
adresse$ : 6, rue du Chemin-Vert
ville$   : PARIS
code_p$  : 75014
pays$    : FRANCE
tél$     : (1) 727.67.90
```

Nous disposons donc de certains éléments qui vont nous être utiles ; le téléphone : remarquez que c'est un champ de type alphanumérique dans lequel figurent les indicatifs et des chiffres séparés par des points. Il va donc falloir repérer dans quel

département la personne habite, et ajouter un 4 si celui-ci correspond à l'indicatif actuel (1). Dans les autres départements, il suffira de supprimer les parenthèses qui séparent l'indicatif. Auparavant, il aura fallu écarter tous les enregistrements des personnes qui habitent à l'étranger et pour lesquelles une telle transformation ne s'impose pas.

```
proc convert
mode 0
éponge
saisis au 5,0; "QUEL FICHIER VOULEZ-VOUS CONVERTIR ?";fich$
ouvre fich$
isole PAYS$="france"
écris au 7,10;"IL Y A";nombenr();"ENREGISTREMENTS QUI SONT
EN FRANCE"
écris au 9,10;"TAPEZ SUR UNE TOUCHE POUR COMMENCER"
que x$=clavier()
éponge
partout
  si souschn(tel$, "(1)")
    que tél$="4"+tél$(5 jusq 5)+"."+tél$(6 jusq long(tél$))
    change
    écris "NOM : ";nom$; tab 20; "TÉL : ";tél$
    fsi
  si souschn (tél$(1 jusq 1),"(") et souschn(tél$(3 jusq 3),
  ")")
    que tél$=tél$(2 jusq 2)+tél$(5 jusq 5)+"."+tél$(6 jusq
    long(tél$))
    change
    écris "NOM : ";nom$; tab 20; "TÉL : ";tél$
    fsi
  si souschn(tél$(1 jusq 1),"(") et souschn (tél$(4 jusq 4),
  ")")
    que tél$=tél$(2 jusq 3)+"."+tél$(6 jusq long(tél$))
    change
    écris "NOM : ";nom$; tab 20; "TÉL : ";tél$
    fsi
  fpartout
fproc
```

Les deux premières lignes de cette procédure sont désormais connues — il est toujours préférable de choisir le mode 0 et de nettoyer l'écran avec **éponge**. Notez la troisième ligne : **saisis** est suivi d'auxiliaires d'impression qui ont pour but d'inscrire le

message qui suit au milieu de l'écran. La variable **fich$** sera affectée du nom du fichier que vous voulez ouvrir pour en convertir les numéros de téléphone. Notez bien que toutes les expressions de cette ligne sont séparées par des point-virgules.

La ligne suivante (**ouvre fich$**) ne présente rien d'extraordinaire ; rappelez-vous que dans ce cas on ne doit pas écrire : **ouvre "fich$"**, qui aurait pour effet d'ouvrir le fichier "fich$", mais **ouvre fich$**, qui ouvre le fichier dont le nom est contenu par **fich$**.

Comme nous l'avions déterminé au départ nous ne conservons que les fiches françaises, grâce à l'instruction **isole**. ARCHIVE affiche alors le nombre d'enregistrements dont le champ Pays$ est égal à FRANCE puis un message indiquant à l'utilisateur qu'il faut taper sur une touche pour amorcer le processus de conversion.

La ligne suivante sert simplement à interrompre l'exécution du programme tant qu'une touche n'est pas frappée. Notez que la variable x$ va être affectée de la touche qui sera tapée, mais cela ne change absolument rien, puisque nous n'utiliserons pas **x$** ultérieurement.

Après avoir nettoyé l'écran une nouvelle fois, la conversion proprement dite commence. Nous commençons par déclarer que nous désirons cette conversion sur l'ensemble du fichier (**partout**) puis trois possibilités sont prévues :

la première est celle des abonnés de la région parisienne qui ont **(1)** comme préfixe, par exemple :

```
( 1 )   5 0 0 . 9 9 . 9 8
| | | | | | | | | | | | |
1 2 3 4 5 6 7 8 9 0 1 2 3
```

La nouvelles numérotation va avoir pour effet de supprimer le **(1)** au profit d'un **4** suivi du restant du numéro. Par conséquent, nous commençons par vérifier que le préfixe est bien **(1)** grâce à l'instruction **souschn()** puis nous donnons l'ordre suivant :
que tel$ soit maintenant égal à "4" suivi du chiffre se trouvant à la position 5 suivi d'un point suivi du reste du numéro de téléphone qui lui ne change pas.

Remarquez comment nous obtenons la fin du numéro de téléphone :

tél$(6 jusq long(tél$))

De la position 6 jusqu'à la fin de la chaîne, puisque **long(tél$)** contient la longueur de *l'ancien numéro*.

Une fois que cette transformation s'est effectuée, nous **échangeons** l'ancien numéro contre le nouveau ; l'instruction **change** est

indispensable ; sans elle la modification ne serait pas prise en compte. Enfin, pour constater la conversion immédiatement, nous demandons à ARCHIVE d'afficher le nom et le numéro de téléphone immédiatement après la transformation.

Le test suivant concerne les abonnés qui n'avaient qu'un seul chiffre comme préfixe et qui est désormais intégré à leur numéro complet. Pour cela nous testons si le premier caractère est une parenthèse "(" et si le troisième est une autre parenthèse ")". Si c'est le cas nous prenons le caractère qui se trouve à l'intérieur, puis nous le collons au reste du numéro en ajoutant bien sûr un point entre les deux premiers chiffres et les suivants, un peu à la manière des habitants de la région parisienne.

Enfin le troisième test concerne les personnes habitant en province et ayant un préfixe à deux chiffres. Selon le même principe, nous testons cette fois le premier et le quatrième caractère et nous ajoutons les chiffres qui se trouvent à l'intérieur des parenthèses.

Enfin nous terminons l'opération (**fpartout**), mais le fichier reste ouvert, afin de visualiser l'effet de cette conversion sur l'ensemble du fichier.

Notez que cette méthode présente l'avantage de ne modifier que les numéros dont la structure était égale à celle que nous avons prévue. Pour les numéros dont les préfixes n'avaient pas été rentrés, aucune conversion n'est opérée.

Vous pouvez imaginer toutes sortes de procédures pour simplifier votre travail — vous savez maintenant comment les créer — et le mieux en matière de programmation est de faire des essais vous-même. Le chapitre suivant vous montrera, au travers d'applications pratiques, comment résoudre certains problèmes.

Pour le moment, si vous n'êtes pas encore très sûr de vos talents de programmateur créez des procédures de plus en plus ardues — il n'y a quasiment pas de problème qui ne soit solvable.

3.5 REMARQUES À PROPOS D'INSTRUCTIONS PARTICULIÈRES

3.5.1 Traitement des erreurs

Nous avons vu dans la section 3.2, l'instruction **erreur** et la fonction **numerr()** qui permettent toutes deux de traiter des erreurs éventuelles. En effet, supposons par exemple que dans une procédure vous demandiez à un utilisateur qui ne connaît rien à ARCHIVE de rechercher un enregistrement dans un fichier alors que ce dernier n'est pas ouvert :

```
proc cherch
saisis; "ENTREZ LE NOM RECHERCHE"; no$
cherche nom$=no$
fproc
```

Or supposons que notre utilisateur ne sache pas que le fichier n'est pas encore ouvert et qu'il tape un nom, comme il lui est demandé. Aussitôt l'exécution du programme s'interrompra pour afficher le message suivant :

```
erreur 94 : fichier non ouvert
```

Désormais, le programme s'est arrêté et notre utilisateur est complètement perdu. Pour remédier à cela, il suffit de créer une nouvelle procédure qui sera chargée d'afficher un message indiquant à l'utilisateur qu'il a fait une erreur et qu'il doit recommencer. Le programme ne s'interrompra pas et l'utilisateur ne sera pas perdu !

Exemple :

```
proc err94
saisis "ENTREZ LE NOM RECHERCHE";no$
si numerr()=94
écris "VOUS DEVEZ AU PREALABLE
OUVRIR UN FICHIER"
sinon
cherche nom$=no$
fsi
fproc
```

Cette procédure devra être appelée à partir de la procédure principale

```
proc cherch
erreur err94
fproc
```

Vous pouvez vous-même faire l'essai, le programme ne s'interrompra pas, il affichera le message d'erreur et reviendra dans la procédure où s'est produit l'erreur. En fait dans le cas présent cela n'offre aucun intérêt puisque le programme ne prévoit pas de sortie possible en particulier pas de possibilité d'ouvrir un fichier. Pour pallier cet inconvénient, nous allons spécifier, dans la procédure **err94**, qu'il faut s'orienter vers une procédure permettant d'ouvrir un fichier. Voici le listing de ces trois procédures :

```
proc err94
saisis "QUEL NOM CHERCHEZ-VOUS";no$
cherche nom$=no$
si numerr()=94
ouvrefich
fsi
fproc

proc ouvrefich
écris "VOUS DEVEZ AU PREALABLE OUVRIR UN FICHIE »
écris
saisis "QUEL FICHIER VOULEZ-VOUS OUVRIR";fich$
ouvre fich$
erreur err94
fproc
```

La procédure que nous venons d'écrire tient compte du numéro d'erreur et donne un message en conséquence. En fait vous pouvez être beaucoup plus paresseux et vous contenter d'écrire quelque chose comme ceci.

```
proc err
saisis "ENTREZ UN NOMBRE",nbr
écris "VOUS AVEZ COMMIS UNE ERREUR No :";numerr()
écris "RECOMMENCEZ"
fproc
```

qui pourra être appelée de n'importe quelle autre procédure

```
proc test
erreur err
fproc
```

Si vous rentrez une lettre à la place d'un chiffre dans la procédure **err**, cette dernière se contentera d'afficher le numéro de l'erreur et de retourner au point de départ. Bien sûr, cela est beaucoup moins explicite, mais beaucoup plus simple.

Pensez à toujours protéger vos applications par des routines d'erreur de ce genre, afin d'éviter à d'éventuels utilisateurs d'être perdus au milieu d'une application — c'est à ce genre de détails que l'on reconnaît un bon programme.

3.5.2 La commande exéc

Cette commande est extrêmement utile pour ceux qui désirent offrir des solutions toutes faites à des utilisateurs qui ne connaissent pas ARCHIVE. Ainsi, si vous concevez une application suffisamment bien faite, il vous suffira d'indiquer, à la personne utilisatrice, deux choses :

```
comment charger ARCHIVE
taper "exéc" suivi d'un nom de programme
```

et le reste se fera tout seul !

exéc a la particularité de charger un programme et de lancer l'exécution d'une procédure nommée **start**. Or, vous savez qu'à partir d'une procédure (en l'occurrence celle-ci) vous pouvez appeler toutes les autres. Il faut donc *obligatoirement* qu'il y ait une procédure appelée **start** pour qu'**exéc** soit efficace. La première application que nous proposons au chapitre 4 vous montrera un exemple pratique.

3.5.3 Les paramètres

Lors de l'appel d'une procédure, vous pouvez lui adjoindre ce que l'on appelle *des paramètres*. Exemple :

```
proc test; x$
écris x$
fproc
```

Lorsque vous appelez cette procédure, il faut spécifier la valeur de **x$**, qui sera alors affichée.

Exemple : test; "Bonjour"

La chaîne "Bonjour" va *passer* comme paramètre à la procédure test, qui l'interprétera comme la valeur que devra prendre **x$**. Cela ne semble peut-être pas évident pour l'instant, mais vous verrez comment l'utiliser dans les chapitres suivants. Sachez d'ores et déjà que vous pouvez spécifier plusieurs paramètres après une procédure :

```
proc test; a$,b$,c$
écris a$;b$;c$
fproc
```

et que, par conséquent, l'appel de cette procédure doit aussi contenir trois paramètres :

```
test; "Bonjour","comment","va"
```

Vous pouvez ainsi définir de véritables fonctions comme par exemple la fonction **carré** ou **cube**.

```
proc carré; x
que resul=x*x
écris "CARRE DE :"; x;"="; resul
fproc
proc cube; x
que resul=x*x*x
écris "CUBE DE :"; x;"=" resul
fproc
```

3.5.4 Les variables locales

Chaque procédure, vous l'avez vu, est indépendante. Il n'en va pas de même pour les variables que nous avons déclarées jusqu'ici. Ces dernières sont ce que l'on appelle des *globales*, c'est-à-dire qu'elles sont connues de tout le programme (par toutes les procédures). ARCHIVE vous offre la possibilité de limiter la connaissance des variables à une procédure seulement. Pour cela, il faut les déclarer *locale*.

Exemple :

```
proc carré; x
local resul
que resul=x*x
écris resul
fproc
proc cube; x
local resul
que resul=x*x*x
écris resul
fproc
```

Nous avons donc repris les procédures précédemment montrées pour expliquer les variables locales. En effet, vous aurez remarqué que les deux procédures plaçaient le résultat qu'elles calculent dans la variable **resul**. Or en les déclarant locales, ces deux variables sont indépendantes et n'existent qu'à l'intérieur des procédures respectives. Cela conduit au constat suivant : si vous

repassez au mode direct, et que vous demandez à afficher **resul**, ARCHIVE répondra par un message d'erreur indiquant que **resul** n'existe pas.

Tout cela est peut-être un peu difficile à comprendre mais sachez que l'utilité des variables locales réside en ce qu'elles sont propres à une procédure et qu'elles ne peuvent par conséquent pas interférer avec le reste d'un programme.

3.5.5 La récursivité

On désigne sous le terme de *récursivité* le fait, pour une procédure, de pouvoir s'appeler elle-même ! Si l'on considère l'exemple suivant :

```
proc saisir
saisis "ENTREZ UN NOMBRE ENTRE 1 ET 5"; X
si x>0 et x<6
écris "OK!"
sinon
saisir
fproc
```

l'effet de cette procédure est de contrôler le nombre qui est entré au clavier ; si celui-ci est dans les limites, l'écran affiche "OK", dans le cas contraire le nombre n'est pas pris en compte et la procédure est rappelée.

3.6 UTILISATION MULTIFICHIERS

Une des particularités d'ARCHIVE est de pouvoir utiliser plusieurs fichiers en même temps. A priori, cela ne présente pas un avantage certain, mais nous allons voir que cela peut être très utile.

Pour pouvoir utiliser ARCHIVE avec plusieurs fichiers ouverts en même temps, il faut lors de la création ou de l'ouverture spécifier un *nom logique de fichier* qui servira à les différencier ultérieurement. En effet deux fichiers peuvent avoir exactement la même structure et pourtant être différents.

Nous avons déjà donné ce conseil précédemment mais répétons-le : les manipulations multifichiers imposent que l'on spécifie les noms logiques de fichier quasiment à chaque opération. Il est par conséquent beaucoup plus rapide de donner des noms logiques le plus court possible afin d'économiser du temps de frappe.

La meilleure façon de se rendre compte de l'utilisation de plusieurs fichiers en même temps est de créer une petite procédure utilisant cette facilité.

Nous allons ajouter des champs supplémentaires au fichier que nous avons créé précédemment. Nous avons dit auparavant qu'il fallait bien prévoir l'usage que l'on allait faire du fichier car il était difficile d'ajouter des champs ultérieurement. En fait la seule solution consiste à créer un nouveau fichier contenant les nouveaux champs et d'y recopier les informations que l'on possède déjà dans le premier. Cette procédure permet de faire ceci :

```
proc champ
éponge
écris "UN INSTANT..."
crée "carnet2" logique "a"
nom$
prénom$
adresse$
ville$
codep$
pays$
tel$
divers$
fcrée
lis "carnet" logique "b"
partout "b"
écris "NOM :";nom$; tab 20;"PRÉNOM : ";prénom$
que a.nom$=b.nom$
que a.prénom$=b.prénom$
que a.tel$=géné(b.tel,8)
ajoute "a"
fpartout
ferme "a"
ferme "b"
fproc
```

Les deux premières lignes de cette procédure sont désormais classiques (indiquez toujours par un message d'attente que l'ordinateur est en train de travailler).

Nous créons le nouveau fichier tel que nous souhaitons maintenant le voir structuré (nous n'avons mis que des champs alphanumériques), n'oubliez pas **fcrée** à la fin, dans une procédure cette instruction est obligatoire.

Puis nous ouvrons le fichier que nous avons déjà créé (**carnet**) uniquement pour la lecture puisque nous n'allons pas faire de modification dans celui-ci. Notez l'instruction **partout "b"**, n'oubliez pas de préciser le nom logique du fichier que vous voulez explorer du début jusqu'à la fin.

Toujours dans le souci de visualiser immédiatement les effets de ce que vous êtes en train de faire, nous demandons que les noms et prénoms s'affichent à l'écran successivement.

Puis nous déclarons que les trois champs du nouveau fichier (du nom logique "a") sont maintenant égaux au trois champs de l'ancien fichier que nous avons créé (de nom logique "b"), puis à chaque fois qu'un enregistrement est lu et recopié dans son ensemble nous l'ajoutons au nouveau fichier (**ajoute "a"**).

Remarquez que pour le téléphone, ayant transformé notre champ numérique en un champ alphanumérique, il faut convertir le chiffre en une chaîne par **gene()** qui transforme un chiffre en une chaîne de caractères selon un format général (la longueur de la chaîne a été fixée à 8, c'est la longueur maximum que prenaient nos numéros de téléphone).

Enfin, nous refermons les deux fichiers.

Le résultat est bien conforme à celui que nous attendions ; le nouveau fichier contient bien tous les noms, prénoms et numéros de téléphone, mais les autres champs sont, bien entendu, vides de toute information puisque celle-ci n'existait pas au départ.

Voilà donc un problème résolu. Vous l'aurez constaté, ce n'est pas compliqué, et **il n'y a pas d'autre moyen que celui-ci pour changer la structure d'un fichier.**

Mais l'usage de plusieurs fichiers ne se limite naturellement pas à ce cas précis ; vous pouvez créer toutes sortes d'applications dans lesquelles plusieurs fichiers sont nécessaires, nous verrons cela dans les chapitres suivants.

ET S'IL N'Y A PLUS DE PLACE ?

La place pour créer un second fichier et faire des opérations de transfert peut rapidement manquer. Notamment sur le QL où la mémoire de masse est beaucoup plus limitée.

Pour remédier à cela, rien ne vous empêche de retirer la disquette ou la cassette du lecteur de gauche (celui qui contient XCHANGE/ARCHIVE normalement) et de la remplacer par une disquette/cassette vierge).

Dans ce cas il faudra remplacer la ligne :

crée "carnet2" logique "a"

Par :

crée "a:carnet2" logique "a" IBM PC

ou :

crée "mdv1_carnet2" logique "a" QL

et le nouveau fichier sera alors créé sur un support offrant de la place.

CHAPITRE 4

Exemples d'applications

Le but de ce chapitre n'est pas de donner une liste exhaustive de toutes les réalisations possibles dans ARCHIVE, mais de montrer quelques exemples qui vous guideront dans l'emploi futur de votre logiciel. Beaucoup trop de gens considèrent qu'un logiciel de gestion de bases de données est exclusivement réservé au traitement de fichiers ; en fait, toutes sortes d'applications sont possibles, même les plus inattendues.

Avant d'aborder ce chapitre il est conseillé de bien avoir assimilé la programmation d'ARCHIVE ; en effet toutes les procédures que nous allons vous montrer font appel à bon nombre de fonctions et d'instructions qui constituent le langage de programmation d'ARCHIVE. Bien sûr, vous pouvez vous contenter de recopier sans comprendre, mais la majorité des applications personnelles que vous aurez à résoudre nécessiteront vraisemblablement une petite adaptation de votre part.

Programmer en ARCHIVE devient un jeu d'enfant, lorsque l'on a bien compris la façon de le faire. Rappelez-vous, fragmentez toujours vos tâches au maximum et prévoyez l'emploi d'ARCHIVE par une personne qui ne le connaît pas !

4.1 QUI SE SERT D'ARCHIVE, QUAND, PENDANT COMBIEN DE TEMPS ?

Dans la majorité des entreprises, seraient-elles petites, il est souvent nécessaire de savoir à quoi le plus de temps est dépensé, voire perdu. Les méthodes d'analyse du temps passé ne sont pas aisées puisqu'elles requièrent un contrôle permanent et par conséquent une perte de temps supplémentaire.

Dans la première application que nous allons décrire, le temps passé, la date et la personne qui aura utilisé ARCHIVE, seront régulièrement enregistrés dans un fichier à part, que nous avons appelé du nom fort peu élégant de *mouchard*. En fait, s'il s'agit bien d'un *mouchard* à proprement parler, c'est surtout une aide bien utile au contrôle de l'utilisation d'ARCHIVE !

4.1.1 Le fichier « mouchard »

Pour rendre ce fichier annexe opérant, il faudra auparavant créer un fichier de ce nom dont les champs devront être :

```
numéro$
dateutil$
hdébut$
hfin$
```

Ce fichier ne sera ouvert qu'à la fin d'une session, ce qui signifie qu'il ne devrait pas interférer avec le fichier sur lequel l'utilisateur aura travaillé. Dans ce fichier seront consignées les informations suivantes :

numéro$: sera le numéro utilisateur de la personne qui aura utilisé ARCHIVE. Il va donc de soi, que chaque utilisateur devra avoir son propre numéro.

dateutil$: sera la date à laquelle ARCHIVE aura été utilisé par cette personne, il sera ainsi facile de retrouver une erreur, par exemple.

hdébut$ sera chargé d'enregistrer à quel moment l'utilisation d'ARCHIVE aura débuté. A noter que cette heure ne correspond pas nécessairement à l'heure réelle d'utilisation, elle sert simplement de repère de temps d'utilisation puisque nous pourrons la comparer à :

hfin$ contiendra l'heure de fin de session, et permettra donc les statistiques sur le temps passé devant ARCHIVE.

Ces informations enregistrées, ce fichier est refermé et l'utilisateur ne s'en aperçoit pas.

```
proc quit
  éponge
  écris au 7,20;"UN INSTANT..."
  ferme
  ouvre "mouchard"
  que numéro$=numutil$
  que dateutil$=date$
  que hdébut$=heure$
  que hfin$=heure()
  ajoute
  ferme
  écris au 9,20;"TERMINE "
  fproc
```

4.1.2 Un accès limité

Il peut aussi arriver que l'on veuille limiter l'accès de certaines données à un nombre restreint de personnes. On a la plupart du temps recours au banal mot de passe ; banal, mais efficace !

La procédure **passe** que nous montrons ci-après ne livrera les secrets d'ARCHIVE que si le mot de passe *sésame* est entré. Si après trois tentatives ce mot n'est pas trouvé, ARCHIVE est abandonné, de quoi décourager les indésirables...

Bien entendu ce mot de passe peut être changé comme vous le voulez. Une fois ce mot de passe modifié, sauvegardez votre programme en le protégeant :

sauve protégé nom-de-programme

Lorsque le programme est sauvegardé de cette manière, il est impossible d'y avoir accès (pas de listing, pas de sauvegarde, etc.). Attention, cela vaut aussi pour vous ! Prenez donc garde à toujours conserver sur une disquette à part une version non protégée, pour d'éventuelles modifications futures.

N.B. : Un programme sauvegardé en version protégée est automatiquement sauvé en programme objet (code binaire) et devra donc être rechargé de cette façon. Cette possibilité n'est offerte que sur l'IBM PC.

Exemple : charge objet "nom-de-programme"

LISTING DES PROCÉDURES

```
proc start
  que x=0
  mode 0
  éponge
  écris au 5,10;"BIENVENUE DANS ARCHIVE"
  tempo;20
  saisis au 7,10;"ENTREZ VOTRE NUMERO UTILISATEUR ";numutil$
  éponge
  écris au 5,10;"LA DATE D'AUJOURD'HUI EST-ELLE (O/N) ";date(1);
  que rep$=majus(clavier())
  si rep$<>"O"
    saisis au 7,10;"ENTREZ LA DATE CORRECTE ";date$
    sinon
    que date$=date(1)
    fsi
  que heure$=heure()
  éponge
  passe
  proc1
  fproc
```

Rappelez-vous que vous avez la possibilité de faire démarrer un programme automatiquement en nommant une procédure **start**, qui pourra à son tour en appeler d'autres. Cette possibilité est particulièrement adaptée pour les personnes ne connaissant pas la manipulation d'ARCHIVE ; dans le cas présent, il suffit d'indiquer comment rentrer dans ARCHIVE, puis de taper :

exéc objet "nom-de-programme"

Aussitôt ARCHIVE charge le bon programme et commence à exécuter la procédure **start** qui sera généralement un menu des diverses possibilités offertes.

Dans le cas présent un message de bienvenue est affiché et le numéro d'utilisateur demandé. Ensuite la date de l'horloge interne de l'ordinateur est affichée, précédée d'un message demandant à l'utilisateur si cette date est correcte. Dans le cas contraire la date correcte est demandée. Cette opération est particulièrement utile ici, puisque nous allons en fin de session enregistrer cette information. Dans la majorité des cas, il est prudent de le faire, car les opérations qui demandent le maniement des dates sont assez courantes.

Notez que, quelle que soit la date prise en compte, c'est toujours

la variable **date$** qui est affectée ; toutes les opérations sur les dates devront donc désormais faire référence à cette variable.

Comme nous l'indiquions, l'heure de l'horloge interne est enregistrée, qu'elle soit exacte ou non. On se dirige alors vers la procédure **passe**.

```
proc passe
  éponge
  saisis au 5,10;"ENTREZ VOTRE MOT DE PASSE ";pass$
  si pass$="SESAME" ou pass$="sesame"
    reviens
    sinon
    si x<3
      que x=x+1
      passe
      sinon
      écris au 7,10;"VOUS AVEZ ECHOUE APRES TROIS TENTATIVES"
      tempo;20
      note quitte
      fsi
    fsi
  fproc
```

Rien de bien compliqué pour cette procédure ; remarquez la façon de faire des boucles avec **tantque/ftantque**, qui peut sembler étrange aux personnes qui ont l'habitude de programmer en Basic. En fait cette instruction est très pratique et beaucoup plus souple d'emploi qu'une boucle FOR...NEXT où l'argument suivant FOR ne peut généralement pas être modifié au sein de la boucle alors que **tantque/ftantque** le permet.

C'est bien sûr à la troisième ligne que vous devrez remplacer « SESAME » par un autre mot de votre choix si vous souhaitez modifier le mot de passe. Lorsque le mot de passe est correctement entré, le programme revient à la procédure d'appel (**start**) et est immédiatement orienté vers **proc1** ; nous y reviendrons. Si le mot de passe au bout de trois fois n'est pas détecté, on quitte ARCHIVE, comme nous l'avons déjà indiqué, mais auparavant un message apparaît à l'écran indiquant les erreurs successives de l'utilisateur :

« VOUS AVEZ ECHOUE APRES TROIS TENTATIVES »

Notez au passage la façon de faire apparaître ce message pendant un certain temps avant de tout quitter. Nous avons en effet recours à une procédure de temporisation (**tempo**) avec un passage de paramètre pour contrôler la durée de cette temporisation. Nous utiliserons souvent ce genre de procédure pour

permettre à l'utilisateur d'avoir suffisamment de temps pour lire un message à l'écran.

```
proc tempo;tp
  local t: que t=0
  tantque t<tp
    que t=t+1
    ftantque
  fproc
```

Cette procédure n'a aucun autre intérêt que celui de ralentir l'exécution d'un programme pendant un certain temps indiqué dans le paramètre **tp**.

```
proc proc1
  éponge
  écris "LA DATE D'AUJOURD'HUI EST : ";date$
  écris "VOUS ETES L'UTILISATEUR : ";numutil$
  écris
  écris "VOICI LE MENU DES OPERATIONS DISPONIBLES"
  tempo;50
  menu
  fproc
```

Cette procédure est l'enchaînement logique de ce que nous venons de voir jusqu'ici. Tous contrôles passés et toutes informations consignées, il ne reste plus à l'utilisateur qu'à indiquer quelle opération il veut réaliser. Après un bref récapitulatif de l'identité de l'utilisateur, la procédure **menu** est appelée.

```
proc menu
  éponge
  écris au 2,40;"MENU"
  écris au 6,20;"A - OUVRIR UN FICHIER"
  écris au 7,20;"B - FERMER UN FICHIER"
  écris au 8,20;"C - RECHERCHER UN ENREGISTREMENT"
  écris au 9,20;"D - FAIRE UNE MODIFICATION"
  écris au 10,20;"E - AJOUTER UN ENREGISTREMENT"
  écris au 11,20;"F - SUPPRIMER UN ENREGISTREMENT"
  écris au 12,20;"G - AFFICHER LE CATALOGUE"
  écris au 13,20;"H - RETOURNER AU MODE DIRECT"
  écris au 15,20;"VOTRE CHOIX : ";
  que chx$=majus(clavier())
  si chx$="A":proca: fsi
  si chx$="B":procb: fsi
  si chx$="C":procc: fsi
  si chx$="D":procd: fsi
  si chx$="E":proce: fsi
  si chx$="F":procf: fsi
  si chx$="G":procg: fsi
  si chx$="H":proch: fsi
```

```
écris au 15,20;"CE CHOIX N'EXISTE PAS - RECOMMENCEZ"
tempo;30
menu
fproc
```

Nous avons sciemment « fabriqué » un menu important, pour vous montrer comment bien orienter un utilisateur. Ici, seule la première option est disponible (ouvrir un fichier), mais la suite est facilement concevable. Remarquez la façon d'indiquer qu'un choix est erroné, toujours en faisant appel à la procédure de temporisation (**tempo;30**) et en rappelant le menu en cas de choix illicite (grâce à la récursion). Supposons donc que vous désiriez ouvrir un fichier (choix A) ; la procédure **proca** est aussitôt sélectionnée.

```
proc proca
  éponge
  erreur err100
  si numerr()=100
    écris tab 20;"CE FICHIER N'EXISTE PAS -- ERREUR -- RECOMMENCEZ"
    tempo;20
    éponge
    erreur err100
    fsi
  que nbrfich=nbrfich+1
  fproc
proc err100
  saisis au 7,10;"ENTREZ LE NOM DU FICHIER QUE VOUS VOULEZ OUVRIR ";fich$
  si nbrfich>1
    saisis au 12,20;"QUEL NOM LOGIQUE ? ";nl$
    sinon
    que nl$="maître"
    fsi
  lis fich$ logique nl$
  fproc
```

Notez dans **menu**, que lorsque l'on demande un choix à l'utilisateur, sa réponse est toujours transformée en majuscule. Cela évite qu'un choix ne soit pas reconnu, simplement parce que la *casse* n'est pas la bonne. Si la variable **nbrfich**, qui contient le nombre de fichiers ouverts en même temps, est supérieure à 1, un choix de nom logique est proposé. Au cas où aucun nom logique n'est spécifié, celui-ci est *par défaut* « maître ».

Vous aurez sans doute remarqué la façon d'ouvrir le fichier au sein d'une procédure **err100**. Celle-ci permet d'éviter l'interruption de l'exécution du programme au cas où la bonne disquette/cassette n'aurait pas été mise dans le lecteur par défaut, ou toute autre erreur similaire. Pensez à mettre des procédures de ce genre lorsque vous concevez vos propres applications.

Pour terminer, le programme est réorienté vers le menu des commandes disponibles et il ne vous reste plus qu'à l'adapter à vos besoins...

4.2 UNE APPLICATION COMPLÈTE POUR UN PROBLÈME QUOTIDIEN

Les comptes personnels, aussi simples soient-ils, posent toujours un problème de rangement, de classement, sans parler bien sûr des recherches postérieures en cas de litige. Encore une fois ARCHIVE va nous permettre de nous y retrouver de façon beaucoup plus claire, et surtout, calculer pour nous de façon très précise.

A nouveau, le programme est articulé autour d'un menu où différentes opérations sont possibles :

— enregistrer des opérations
— fermer le fichier et abandonner le programme
— analyser les dépenses et les revenus par type
— rechercher un enregistrement.

C'est donc une application complète d'une comptabilité personnelle que nous allons décortiquer pour vous.

En premier lieu, le fichier que nous allons utiliser devra être créé. Un seul fichier pour les débits et les crédits est utilisé, puisque l'utilisateur spécifie à chaque fois la nature de l'opération. Nous avons structuré notre fichier de la façon suivante :

```
créé
montant
libellé$
date$
code$
nature$
mod$
numéro$
solde
fcrée
```

Les informations contenues par chacun des champs seront les suivantes :

montant : le montant de l'opération en francs
libellé$: une ligne de commentaire sur l'objet de l'opération
date$: la date de l'opération

code$: le code de l'opération, qui servira pour les statistiques
nature$: C pour un crédit, D pour un débit
mod$: le mode de transaction (chèque, carte de crédit, etc.)
numéro$: par exemple le numéro d'un chèque ou d'un virement
solde : mise à jour permanente du solde au fur et à mesure

En outre nous avons dessiné un *masque d'écran*, représenté ici. Ce n'est pas un format obligatoire, vous pouvez le modifier à votre convenance.

MONTANT : 100	**DATE :** 01/01/86
CODE : 1	**NATURE :** D
NUMERO : 606	
LIBELLE : Vacances été	
MODE DE PAIEMENT : Chèque	**SOLDE :** 8779

Figure 4.1 : *Masque d'écran pour la comptabilité personnelle.*

```
proc start
  erreur ouvrir
  si numerr()>0
    écris "ERREUR DE LECTURE - VERIFIER VOTRE SUPPORT"
    que t$=clavier()
    start
    fsi
  menu
  fproc
```

Encore une fois le programme commence par une procédure de nom **start** afin de simplifier l'utilisation par le chargement automatique. Cette procédure fait aussitôt appel à une routine de traitement d'erreur au cas où le support ne contiendrait pas le fichier **compte** ou serait absent, comme précédemment décrit.

```
proc ouvrir
  ouvre "compte"
  chargem "compte"
  fproc
```

Notez que le fichier et le masque d'écran portent le même nom, mais que par le jeu des extensions (cf. chapitre 3) ils ne sont pas confondus. Si tout se passe correctement, l'utilisateur a immédiatement accès au menu.

```
proc menu
  mode 0
  écris au 4,30;"MENU"
  écris au 6,20;"A - ENREGISTRER DES OPERATIONS"
  écris au 8,20;"B - FERMER ET QUITTER"
  écris au 10,20;"C - ANALYSER LES DEPENSES"
  écris au 12,20;"D - RECHERCHE"
  écris au 16,20;"VOTRE CHOIX ";
  que chx$=majus(clavier())
  si chx$="A": fin :maz:oper: fsi
  si chx$="B":quit: fsi
  si chx$="C":analyse: fsi
  si chx$="D":recher: fsi
  menu
  fproc
```

Menu désormais classique, au sein duquel sont réunies toutes les possibilités énoncées plus haut.

Choix A :

Le premier choix (enregistrer des opérations) oriente sur deux procédures. Le mot **fin** n'est pas un nom de procédure mais la commande qui sert à appeler le dernier enregistrement du fichier (nécessaire, car c'est lui qui contient le dernier solde). La première :

```
proc maz
  que montant=0
  que libellé$=""
  que date$=""
  que code$=""
  que nature$=""
  que mod$=""
  que numéro$=""
  fproc
```

maz signifie **mise à zéro** ; comme vous pouvez le voir un certain nombre de champs sont « vidés » de leur contenu. Attention, cela ne signifie pas que l'on efface le contenu de certains champs dans le fichier. Ceci est un point très important : les champs que nous remettons à zéro ne le sont que *dans la mémoire de l'ordinateur* ; si nous tapions maintenant **ajoute**, là seulement l'enregistrement ne contiendrait que des champs vidés de leur contenu. Mais ce n'est pas ce que nous faisons ici, nous nous contentons de *préparer* les variables/champs pour recevoir des nouvelles informations, qui, elles, seront **ajoutées** au fichier.

Pourquoi cette opération ? La raison en est simplement que, si vous effectuez plusieurs opérations de suite, les variables/champs n'étant pas réinitialisées à zéro à chaque fois, le simple fait de

taper <RETURN>, réenregistre à nouveau l'information précédente. Vous pouvez faire le test vous-même, pour vous rendre exactement compte de l'effet produit ; il suffit de retirer **maz** de la ligne qui correspond au choix A.

Cette opération de remise à zéro effectuée, on se branche immédiatement vers la procédure qui permet d'entrer des enregistrements.

```
proc oper
  éponge
  écran
  montre
  écris au 21,2;"ENTREZ LE MONTANT"
  saisism montant
  écris au 21,2;"ENTREZ LA DATE (JJ/MM/AA)"
  saisism date$
  écris au 21,2;"ENTREZ LE CODE (1 - FOYER, 2 - VOITURE, 3 - LOISIRS, ETC.)"
  saisism code$
  écris au 21,2;"ENTREZ LE NUMERO DE L'OPERATION (DU CHEQUE, DU VIREMENT ETC..."
  saisism numéro$
  écris au 21,2;"ENTREZ LA NATURE (C pour CREDIT, D pour DEBIT)                "
  saisism nature$
  écris au 21,2;"ENTREZ LE MODE DE PAIEMENT                                    "
  saisism mod$
  écris au 21,2;"ENTREZ LE LIBELLE                                             "
  saisism libellé$
  écris au 21,2;"CES INFORMATIONS SONT ELLES EXACTES (O/N)"
  que rep$=majus(clavier())
  si rep$="O"
    si majus(nature$)="C"
      que solde=solde+montant
      sinon
      que solde=solde-montant
      fsi
    montre
    ajoute
    écris au 21,2;"TAPEZ C POUR UNE AUTRE OPERATION ET M POUR RETOURNER AU MENU"
    que rip$=majus(clavier())
    si rip$="C"
      fin
      maz
      oper
      sinon
      menu
      fsi
    sinon
    oper
    fsi
  fproc
```

Cette procédure est celle qui vous permet de rentrer les opérations. Chaque information à rentrer est accompagnée d'un message guide qui s'affiche en bas de l'écran. Remarquez la façon dont sont saisies les informations. C'est en effet la fonction **saisism** qui est utilisée. Cette fonction permet de se positionner directement en face du champ auquel elle correspond et par conséquent de faciliter l'opération de saisie. N'oubliez pas, pous l'utiliser, il faut forcément que le masque d'écran soit au préalable affiché (**écran** et **montre**).

La suite des opérations est contrôlée par une imbrication de *si sinon fsi*. En effet, nous avons donné la possibilité de revenir sur l'enregistrement si l'utilisateur se rend compte qu'il a fait une faute de saisie. Dans ce cas, les variables/champs ne sont pas remises à zéro, l'enregistrement n'est pas **ajouté** au fichier et il réapparaît à l'écran, permettant ainsi de faire des corrections. Suivant le principe énoncé précédemment, il n'est pas nécessaire de tout retaper. Le simple fait d'appuyer sur <RETURN> à chaque information valable, a pour effet de la valider ; il suffit de corriger l'information sur laquelle on a fait une faute.

Lorsque l'utilisateur signale que les informations sont exactes, l'enregistrement est **ajouté** au fichier, et l'utilisateur peut continuer à enregistrer des opérations (C) ou retourner au menu. Les variables/champs sont remises à zéro et la procédure habituelle continue.

Option C du menu — analyse des données

```
proc analyse
  éponge
  que tot1=0: que tot2=0: que tot3=0: que tot4=0: que tot5=0
  que tot6=0: que tot7=0: que tot8=0: que tot9=0: que tot10=0
  saisis au 5,10;"DEPUIS QUELLE DATE DESIREZ VOUS UNE ANALYSE ";
  saisis dat$
  partout
    si date$>dat$ et nature$="D"
      si code$="1": que tot1=tot1+montant: fsi
      si code$="2": que tot2=tot2+montant: fsi
      si code$="3": que tot3=tot3+montant: fsi
      si code$="4": que tot4=tot4+montant: fsi
      si code$="5": que tot5=tot5+montant: fsi
      fsi
    si date$>dat$ et nature$="C"
      si code$="6": que tot6=tot6+montant: fsi
      si code$="7": que tot7=tot7+montant: fsi
      si code$="8": que tot8=tot8+montant: fsi
      si code$="9": que tot9=tot9+montant: fsi
      si code$="10": que tot10=tot10+montant: fsi
      fsi
```

```
  fpartout
que total=tot1+tot2+tot3+tot4+tot5
que total2=tot6+tot7+tot8+tot9+tot10
éponge
écris au 3,0;repro("-",80)
écris au 4,30;"ANALYSE"
écris au 5,0;repro("-",80)
écris
écris
écris "DEPENSES CODE 1 : ";tot1; tab 30;
si tot1<>0: écris rond(tot1/total*100,5);"%": sinon : écris "0%": fsi
écris "DEPENSES CODE 2 : ";tot2; tab 30;
si tot2<>0: écris rond(tot2/total*100,5);"%": sinon : écris "0%": fsi
écris "DEPENSES CODE 3 : ";tot3; tab 30;
si tot3<>0: écris rond(tot3/total*100,5);"%": sinon : écris "0%": fsi
écris "DEPENSES CODE 4 : ";tot4; tab 30;
si tot4<>0: écris rond(tot4/total*100,5);"%": sinon : écris "0%": fsi
écris "DEPENSES CODE 5 : ";tot5; tab 30;
si tot5<>0: écris rond(tot5/total*100,5);"%": sinon : écris "0%": fsi
écris
écris "REVENU CODE 6 : ";tot6; tab 30;
si tot6<>0: écris rond(tot6/total2*100,5);"%": sinon : écris "0%": fsi
écris "REVENU CODE 7 : ";tot7; tab 30;
si tot7<>0: écris rond(tot7/total2*100,5);"%": sinon : écris "0%": fsi
écris "REVENU CODE 8 : ";tot8; tab 30;
si tot8<>0: écris rond(tot8/total2*100,5);"%": sinon : écris "0%": fsi
écris "REVENU CODE 9 : ";tot9; tab 30;
si tot9<>0: écris rond(tot9/total2*100,5);"%": sinon : écris "0%": fsi
écris "REVENU CODE 10 : ";tot10; tab 30;
si tot10<>0: écris rond(tot10/total2*100,5);"%": sinon : écris "0%": fsi
écris au 21,2;"APPUYEZ SUR UNE TOUCHE POUR REVENIR AU MENU";
que x$=clavier()
menu
fproc
```

Cette procédure représente typiquement ce qu'ARCHIVE peut permettre de réaliser. Nous avons arbitrairement choisi 10 codes (1 à 10). Il va de soi que ceci peut être modifié selon que l'on désire une analyse plus détaillée ou non. Les codes 1 à 5 sont réservés aux dépenses, et les codes 6 à 10 aux revenus. Vous pouvez par conséquent établir une bonne fois pour toutes des codes en fonction de vos dépenses et les répertorier ainsi.

Vous saurez par exemple que le code 1 est pour les dépenses du foyer, le code 2 pour la voiture, le code 6 pour le salaire et le 7 pour un autre revenu quelconque. Ceci établi, vous devrez rentrer un code pour chacunes de vos dépenses et revenus. Lorsque vous choisissez l'option *analyse* de ce programme, vous obtenez en données brutes et en pourcentages la part dépensée ou perçue

pour chacun des codes. Une solution idéale pour savoir où passe l'argent !

Notez que les variables sont toutes initialisées à zéro au départ ; en ARCHIVE les variables ne le sont pas automatiquement. Dans la partie qui écris les résultats à l'écran, nous avons pris la précaution d'ajouter la condition **si total>0** afin d'éviter des erreurs possibles de division par 0. Les résultats sont exprimés en pourcentages arrondis, grâce à la fonction **rond()**.

Cette procédure est un bon exemple du traitement de l'information d'un fichier, vous aurez souvent à vous en servir.

Option D du menu — recherche d'un enregistrement

```
proc recher
  éponge
  écris au 5,0;"DESIREZ VOUS FAIRE VOTRE RECHERCHE APRES UNE DATE (O/N)";
  que rep$=majus(clavier())
  si rep$="O"
    saisis au 7,10;"ENTREZ LA DATE : ";datelim$
    écris au 9,10;"UN INSTANT..."
    isole date$>datelim$
    fsi
  éponge
  écris au 5,0;"ENTREZ (N) POUR UNE RECHERCHE PAR LES NUMEROS"
  écris au 6,0;"ENTREZ (L) POUR UNE RECHERCHE SUR LES LIBELLES"
  écris au 7,0;"ENTREZ (M) POUR UNE RECHERCHE PAR LES MONTANTS"
  que rip$=majus(clavier())
  si rip$="N":numer: fsi
  si rip$="L":libel: fsi
  si rip$="M":mont: fsi
  recher
  fproc
```

Cette procédure vous offre la possibilité de restreindre votre recherche à des enregistrements postérieurs à une date que vous aurez choisie. Notez au passage le message « UN INSTANT » ; il est très important d'indiquer à l'utilisateur qu'il aura à attendre quelque temps (le délai nécessaire à l'**isolement**) de certains enregistrements. Cette procédure propose à nouveau trois choix :

option N du menu — recherche

```
proc numer
  éponge
  saisis au 5,0;"ENTREZ LE NUMERO DE L'OPERATION QUE VOUS RECHERCHEZ ";num$
  cherche num$=numéro$
  tantque vu()
    écran
    montre
    écris au 20,0;"EST-CE CELUI-CI QUE VOUS RECHERCHEZ (O/N) ?";
```

```
  que rep$=majus(clavier())
  si rep$="O"
    smenu
    sinon
    continue
    fsi
  ftantque
écris au 20,0;"IL N'Y A PAS D'ENREGISTREMENT REPONDANT A CE CRITERE"
tempo;20
écris au 20,0;"(R)ETOUR MENU, (A)UTRE RECHERCHE
que rip$=majus(clavier())
si rip$="R":menu: fsi
si rip$="A":recher: fsi
menu
fproc
```

Cette procédure recherche les enregistrements par les numéros (exemple : n° de chèque). Notez la façon de rechercher plusieurs enregistrements au cas où celui qui est proposé n'est pas recherché.

Option L du menu — recherche

```
proc libel
  éponge
  saisis au 5,0;"ENTREZ LE LIBELLE DE L'OPERATION QUE VOUS RECHERCHEZ ";lib$
  cherche souschn(libellé$,lib$)>0
  tantque vu()
    écran
    montre
    écris au 20,0;"EST-CE CELUI-CI QUE VOUS RECHERCHEZ (O/N) ?";
    que rep$=majus(clavier())
    si rep$="O"
      smenu
      sinon
      continue
      fsi
    ftantque
  écris au 20,0;"IL N'Y A PAS D'ENREGISTREMENT REPONDANT A CE CRITERE"
  tempo;20
  écris au 20,0;"(R)ETOUR MENU, (A)UTRE RECHERCHE
  que rip$=majus(clavier())
  si rip$="R":menu: fsi
  si rip$="A":recher: fsi
  menu
  fproc
```

Même type de procédure, si ce n'est l'emploi de la fonction **souschn()**, qui permet de retrouver un morceau de chaîne dans une autre. La particularité ici est de l'utiliser sur un champ seulement

(a la différence de **trouve**, que nous aurions aussi pu employer, mais qui fait une recherche sur tous les champs).

Option M du menu — recherche

```
proc mont
  éponge
  saisis au 5,0;"ENTREZ LE MONTANT DE L'OPERATION QUE VOUS RECHERCHEZ ";mt
  cherche montant=mt
  tantque vu()
    écran
    montre
    écris au 20,0;"EST-CE CELUI-CI QUE VOUS RECHERCHEZ (O/N) ?";
    que rep$=majus(clavier())
    si rep$="O"
      smenu
      sinon
      continue
      fsi
    ftantque
  écris au 20,0;"IL N'Y A PAS D'ENREGISTREMENT REPONDANT A CE CRITERE"
  tempo;20
  écris au 20,0;"(R)ETOUR MENU, (A)UTRE RECHERCHE
  que rip$=majus(clavier())
  si rip$="R":menu: fsi
  si rip$="A":recher: fsi
  menu
  fproc
```

Même principe, mais pour une recherche sur les montants.

A noter la dernière procédure, qui offre toujours les mêmes choix, lorsque l'on a retrouver l'enregistrement que l'on désirait :

```
proc smenu
  écris au 20,0;"(M)ODIFIER, (R)ETOUR AU MENU"
  que rep$=majus(clavier())
  si rep$="M":oper: fsi
  si rep$="R":menu: fsi
  fproc
```

Cette procédure (sous-menu), est appelée systématiquement à la fin de chaque procédure de recherche et donne la possibilité de faire une modification (proc **oper**) ou de retourner au menu.

Enfin, pour terminer, et de façon *indispensable,* la procédure **quit**, qui referme le fichier en cours. Si le fichier n'était pas refermé, toute l'information serait irrémédiablement perdue.

```
proc quit
  éponge
  écris au 10,2;"SOLDE FINAL :";solde;
  ferme
```

```
  écris au 12,2;"A BIENTOT";
  quitte
  fproc
```

Cette application peut rendre de nombreux services, même dans la comptabilité d'un foyer, au demeurant simple, mais demandant beaucoup de rigueur.

```
proc tempo;t
  que x=t
  tantque x>0
    que x=x-1
    ftantque
  fproc
```

CHAPITRE 5

Une application originale

5.1 QUESTIONNAIRE ET DÉPOUILLEMENT SUR ARCHIVE

Si ARCHIVE est capable d'enregistrer des informations et de les redonner aussi facilement, pourquoi ne pas s'en servir pour un sondage.

Ce dernier nécessite en effet de lourdes et peu souples opérations de dépouillement qui font perdre énormément de temps. Avec ARCHIVE, étant donné qu'il est très facile de sélectionner un certain type d'information, les questions croisées, les analyses de toutes sortes, et surtout le dépouillement des questions ouvertes devient un jeu d'enfant.

L'application que nous allons décrire ici permet de dépouiller les résultats d'un questionnaire sous de nombreuses formes. Nous n'avons certainement pas envisagé tous les cas de dépouillement, mais une majorité des possibilités les plus courantes est couverte.

En outre, puisque ARCHIVE est une base de données entièrement programmable, nous ne nous sommes pas limités au dépouillement, mais aussi au questionnaire en lui-même. En effet, pourquoi — dans des cas simples, comme les enquêtes par téléphone — ne pas entrer directement les informations recueillies sur l'ordinateur et éviter ainsi le transfert de l'information.

Ainsi, il devient possible en une seule opération de :

— rédiger le questionnaire et l'imprimer
— saisir l'information au fur et à mesure des réponses
— analyser les réponses immédiatement.

Bien sûr, la constitution du questionnaire et sa programmation suppose que l'on connaisse ARCHIVE relativement bien, mais une fois fait, le questionnaire devient exploitable par n'importe quel enquêteur.

Nous allons, au cours de ce chapitre, passer en revue toutes les procédures qui composent notre (petit) questionnaire, afin de vous donner un éventail de tout ce qu'il est possible de réaliser. Encore une fois, bien sûr, les routines que nous donnons devront vraisemblablement être adaptées à votre cas particulier, mais l'idée générale vous guidera facilement.

5.1.1 Les données du problème

Nous devons faire passer un questionnaire de 8 questions à une centaine de personnes. Ce questionnaire comporte toutes sortes de questions comme vous pouvez le voir ci-joint. Le but du traitement par l'informatique est de saisir l'information le plus rapidement possible et surtout de dépouiller les réponses en un temps record.

5.2 LE QUESTIONNAIRE

5.2.1 La solution informatique

Nous allons, dans un premier temps, concevoir les questions une à une, en ayant soin de donner des indications à l'enquêteur ; chaque question fera l'objet d'une procédure nommée **quest1, quest2, quest3**, etc. Dans un deuxième temps, nous allons écrire les différentes procédures destinées à dépouiller le questionnaire.

Le questionnaire devra être codé, un peu de la même façon que les questionnaires classiques, c'est-à-dire que les informations recueillies seront enregistrées dans des champs, ceux-ci constituant la base de données qui contiendra le tout. La seule contrainte sera de bien noter à part quelles informations se trouvent dans quels champs.

En outre la base de données devra être créée avant de commencer les enregistrements. Un principe de base est que tous les champs seront de type alphanumérique, même ceux destinés à recevoir des chiffres.

Afin de simplifier les opérations, nous conseillons d'avoir des

procédures toutes prêtes en fonction de la quantité nécessaire à chaque enquête.

Par exemple :

une procédure qui crée un fichier de 25 champs
50 champs
75 champs
100 champs
etc.

Dans notre exemple, nous avons appelé nos champs d'une façon très simple :

C1$, C2$, C3$, C4$, C5$, etc...

Si vous avez des procédures toutes prêtes, une fois pour toutes, qui créent à chaque utilisation un fichier aux capacités suffisantes pour votre questionnaire, vous gagnez du temps. Si, par exemple, votre questionnaire utilise 88 champs, vous pouvez très bien utiliser la procédure qui crée 100 champs ; les 12 autres resteront vides de toute information, mais cela ne pose pas de problèmes, d'autant qu'ils ne prendront quasiment pas de place-mémoire.

Un de nos soucis constants va être de prévoir nos procédures le plus largement possible afin qu'elles puissent resservir par la suite, dans d'autres questionnaires.

Comme à l'accoutumée notre programme va s'articuler autour d'un menu.

```
proc menu
  mode 0
  éponge
  écris au 5,20;"MENU"
  écris au 10,5;"1 - ENREGISTRER LE PRECEDENT QUESTIONNAIRE ET PASSER AU SUIVANT"
  écris au 11,5;"2 - PASSER AU SUIVANT SANS ENREGISTRER LE QUESTIONNAIRE PRECEDENT"
  écris au 12,5;"3 - ENTAMER UNE PROCEDURE D'ANALYSE"
  écris au 13,5;"4 - ENREGISTRER QUEST. PRECEDENT ET PROCEDURE D'ANALYSE"
  écris au 14,5;"5 - QUITTER"
  écris au 15,5;"6 - OUVRIR"
  choix;"123456"
  si ch$="1": ajoute :quest1: fsi
  si ch$="2":quest1: fsi
  si ch$="3":analyse: fsi
  si ch$="4": ajoute :analyse: fsi
  si ch$="5": ferme : stoppe : fsi
  si ch$="6"
    saisis au 18,10;"QUEL NOM DE FICHIER ? ";no$
    ouvre no$
    fsi
  menu
  fproc
```

Lorsque vous entrez dans ce programme pour la première fois (en n'ayant pas oublié de créer un fichier aux capacités suffisantes auparavant), il faut sélectionner l'option 6 qui vous permet d'ouvrir le fichier que vous avez créé à l'intention du recueil des données. Cela si vous l'avez refermé après l'avoir créé, évidemment.

Ceci fait, vous repassez au menu et si vous voulez commencer une série d'enquêtes, vous sélectionnez l'option 2 (passer au suivant sans enregistrer le questionnaire précédent).

L'option 2 vous fait passer directement à la première question :

```
proc quest1
  mode 0
  éponge
  écris "QUESTION 1"
  écris
  écris
  écris "POSSEDEZ VOUS OU UTILISEZ VOUS UN MICRO ORDINATEUR ? /C1"
  ouinon
  que C1$=on$
  si C1$="2":suivant: fsi
  quest2
  fproc
```

Remarquez que nous avons inclus le nom du(des) champ(s) concerné(s) dans chaque libellé de question. Cela permet de retrouver facilement au cours de l'analyse le champ à sélectionner pour analyser telle ou telle question (en l'occurrence C1).

La question 1 utilise une procédure que nous avons appelé **ouinon**, et que nous avons mise à part, car elle peut être appelée par n'importe quelle question. Or les questions de ce type sont relativement courantes dans un questionnaire ; elle pourra donc être conservée pour une autre enquête.

```
proc ouinon
  écris au 10,20;"1 - OUI"
  écris au 12,20;"2 - NON"
  écris au 14,20;"3 - NE SAIT PAS"
  écris au 16,20;"REPONSE :";
  que on$=clavier()
  si souschn("123",on$)
    reviens
    sinon
    bis
    ouinon
    fsi
  fproc
```

Cette procédure comprend en fait trois possibilités de réponse (Ne sait pas) ; si ce choix n'a pas lieu d'être dans votre questionnaire, vous pouvez cependant le conserver et ne pas vous préoccuper de la troisième option.

Cette procédure contrôle que le choix qui est fait (1, 2 ou 3) est bien valide, et dans le cas contraire se branche sur une autre procédure que nous avons aussi mise à part, puisqu'elle pourra être appelée à différents endroits du programme. Il s'agit de la procédure **bis**.

```
proc bis
  écris au 20,15;"CHOIX ERRONE - RECOMMENCEZ";
  boucle;10
  écris au 20,15;"                                    "
  fproc
```

Cette procédure n'a pour objet que d'afficher un message indiquant à l'utilisateur qu'il s'est trompé en faisant son choix et qu'il doit recommencer. **Bis** fait à son tour appel à une autre procédure, désormais classique, dont le but est de ralentir l'exécution du programme pour que l'utilisateur puisse lire le message qui vient de s'afficher ; c'est la procédure **boucle**.

```
proc boucle;delai
  local x
  que x=0
  tantque x<delai
    que x=x+1
    ftantque
  fproc
```

Comme à l'accoutumée, boucle est paramétré avec le temps d'attente.

Ces enchaînements effectués, on retourne dans la procédure **ouinon** où l'on est invité à faire un autre choix. Si le choix n'est pas erroné, le retour se fait alors dans la procédure **quest1**. Le premier champ (C1$) est alors affecté de la valeur de on$ (variable de la procédure **ouinon**) et si le choix est "2" (NON), le questionnaire s'arrête là et l'on retourne au menu. Si le choix est "1" (OUI), le questionnaire peut continuer.

```
proc quest2
  éponge
  écris "QUESTION 2"
  écris
  écris "COMBIEN EN POSSEDEZ VOUS ? /C2"
  quant
  si val(qt$)>5
```

```
  que qt$="5"
  fsi
que C2$=qt$
quest3
fproc
```

La question 2 n'offre pas de choix, mais demande une quantité. Nous avons pour ce type de question établi une autre procédure — elle aussi utilisable dans toutes les situations.

```
proc quant
  écris au 21,10;"ENTREZ LE NOMBRE EN QUANTITE ET TAPEZ SUR ENTER ";
  saisis qt$
  reviens
  fproc
```

Dans cette procédure pas de contrôle, puisqu'il ne s'agit pas de choix prédéterminés. En revanche, en retournant dans **quest2**, la quantité est contrôlée et nous avons émis comme condition que si cette dernière était supérieure à 5, elle serait de toute façon égale à 5. Ce genre de cas arrive fréquemment, ici la conversion est automatique. C'est le champ C2$ qui est affecté de la valeur de qt$ (variable de la procédure **quant**).

```
proc quest3
  éponge
  écris "QUESTION 3"
  écris
  écris "L'UTILISATION DE CE MICRO ORDINATEUR EST : /C3"
  écris
  écris au 5,20;"1 - PUREMENT PROFESSIONNELLE"
  écris au 7,20;"2 - AVANT TOUT PROFESSIONNELLE MAIS AUSSI POUR LA DETENTE"
  écris au 9,20;"3 - AUTANT PROFESSIONNELLE QUE POUR LA DETENTE"
  écris au 11,20;"4 - SURTOUT POUR LA DETENTE MAIS EGALEMENT PROFESSIONNELLE"
  écris au 13,20;"5 - SEULEMENT POUR LA DETENTE"
  choix;"12345"
  que C3$=ch$
  si C3$="1"
    comment;"ARRETEZ L'INTERVIEW"
    menu
    fsi
  quest4
  fproc
```

Cette question est d'un type très classique ; faire un choix parmi plusieurs options. Une nouvelle fois, nous en avons fait une procédure séparée qui pourra être appelée dans tous les cas où un choix est nécessaire. Cette procédure est appelée avec un paramètre ; ici **choix;"12345"**. Ceci afin de simplifier les opérations de contrôle, comme nous allons le voir dans la procédure **choix**.

```
proc choix;lim$
  que lm$=lim$
  écris au 19,20;"ENTREZ VOTRE CHOIX ";
  que ch$=majus(clavier())
  si souschn(lim$,ch$)
    reviens
    sinon
    bis
    choix;lm$
    fsi
  fproc
```

Le paramètre passé à **choix** sert à contrôler grâce à la fonction **souschn()** qu'un choix n'est pas erroné, et ceci quel que soit le nombre de choix possibles dans une question. Il suffit de les indiquer lors de l'appel. Le reste du fonctionnement de cette procédure est classique.

Une fois le « filtre » de choix passé, le troisième champ (C3$) est affecté de ch$ (variable retournée par choix). Nous avons mis une condition à la poursuite du questionnaire : si le choix est égal à 1 (Utilisation exclusivement professionnelle), arrêter et passer au suivant. Cela est aussi susceptible d'arriver dans beaucoup de questionnaires. Si le choix 1 est sélectionné, nous faisons appel à une petite procédure — **comment** — dont le rôle est d'afficher en bas de l'écran n'importe quel message passé en paramètre (ici "ARRÊTEZ L'INTERVIEW").

```
proc comment;mes$
  écris au 22,10;mes$
  reviens
  fproc
```

Si le choix n'est pas 1, la question 4 est appelée à son tour.

```
proc quest4
  éponge
  écris "QUESTION 4"
  écris
  écris "L'ORDINATEUR A-T-IL ETE ACHETE PAR : /C4"
  écris au 7,20;"1 - UN HOMME"; au 9,20;"2 - UNE FEMME"
  choix;"12"
  que C4$=ch$
  quest5
  fproc
```

Deuxième utilisation de **choix**, ce qui prouve, s'il en était besoin, qu'elle peut servir à beaucoup de cas ! Ici le paramètre passé est bien sûr "12", puisqu'il n'y a que deux choix. Le champ C4$ est affecté à son tour.

```
proc quest5
  éponge
  écris "QUESTION 5"
  écris
  écris "PARMI LES RAISONS SUIVANTES QUELS SONT LES TROIS MOTIFS PRINCIPAUX"
  écris "QUI VOUS ONT FAIT PRENDRE LA DECISION D'ACHETER UN ORDINATEUR ? /C5"
  écris au 7,10;"1 - J'AI LE SOUCIS DE TOUJOURS ETRE A LA POINTE DU PROGRES"
  écris au 8,10;"2 - PARCE QUE MES AMIS M'ONT INCITE A LE FAIRE"
  écris au 9,10;"3 - PAR GOUT DES NOUVEAUTES"
  écris au 10,10;"4 - POUR APPRENDRE A PROGRAMMER"
  écris au 11,10;"5 - PARCE QUE J'AI VU UNE DEMONSTRATION"
  écris au 12,10;"6 - POUR JOUER"
  écris au 13,10;"7 - AUTRE (PRECISEZ /C6)"
  comment;"TROIS REPONSES AU MAXIMUM"
  multiple;1,7,3
  que C5$=mul$
  si souschn(C5$,"7")
    saisis au 15,10;"PRESISEZ : /C6 ";C6$
    fsi
  quest6
  fproc
```

La question 5 est un exemple de question à choix multiples. Nous n'avons utilisé qu'un champ pour stocker les réponses à cette question et bien sûr une procédure spéciale, qui elle aussi servira dans tous ces cas. Il s'agit de la procédure **multiple**, à qui l'on doit passer trois paramètres : le choix le plus bas (ici 1) le choix le plus haut (ici 7) et le nombre de choix maximum (ici 3).

```
proc multiple;liminf,limsup,max
  local x
  que t=0
  saisis au 19,20;"VOS REPONSES :";mul$
  tantque t<long(mul$)
    que t=t+1
    si val(mul$(t jusq t))<liminf ou val(mul$(t jusq t))>limsup ou long(mul$)>max
      bis
      multiple;liminf,limsup,max
      fsi
    ftantque
  fproc
```

Munie de ses paramètres, **multiple** effectue un contrôle à la fois sur le nombre de choix faits (qui ici ne doit pas être supérieur à 3) et sur le contenu des réponses (s'il n'excède pas les limites (1 et 7)). Quand ce contrôle est effectué, la suite est connue.

N.B. : cette vérification nécessite beaucoup d'opérations pour ARCHIVE et elle peut prendre un délai d'une seconde environ.

Vous aurez noté que l'option 7 impose que vous précisiez la raison. C'est alors le champ C6$ qui reçoit les informations de

cette question ouverte. Nous avons donc maintenant un décalage entre le numéro des champs et des questions (il aura fallu deux champs pour cette seule question 5). C'est pour cette raison que vous devez tenir un état le plus précis possible des champs occupés.

```
proc quest6
  éponge
  écris "QUESTION 6"
  écris
  écris "DISPOSEZ VOUS DU LOGICIEL ARCHIVE ? /C7"
  ouinon
  que C7$=on$
  si on$="1"
    quest7
    sinon
    quest8
    fsi
  fproc
```

Cette question ne présente aucune difficulté, elle permet au contraire selon la réponse, de s'orienter directement vers la bonne question à venir. Ainsi, si la personne répond "2" (NON) ici, il est inutile d'aller à la question 7, mais il faut s'orienter directement vers la question 8. ARCHIVE s'en occupe automatiquement selon la réponse sélectionnée.

```
proc quest7
  éponge
  écris "QUESTION 7"
  écris
  écris "SUR L'ECHELLE SUIVANTE POUVEZ VOUS NOTER LES DEFAUTS ET QUALITES D'ARCHIVE"
  écris au 7,5;"FIABLE"; au 7,70;"PEU FIABLE"
  écris au 8,10;"!------!------!------!------!------!------!------!------!"
  écris au 9,10;"1      2      3      4      5      6      7      8      9"
  écris au 10,5;"RAPIDE"; au 10,70;"LENT"
  écris au 11,10;"!------!------!------!------!------!------!------!------!"
  écris au 12,10;"1      2      3      4      5      6      7      8      9"
  écris au 13,5;"FACILE"; au 12,70;"COMPLIQUE"
  écris au 14,10;"!------!------!------!------!------!------!------!------!"
  écris au 15,10;"1      2      3      4      5      6      7      8      9"
  écris au 16,5;"UTILE"; au 16,70;"INUTILE"
  écris au 17,10;"!------!------!------!------!------!------!------!------!"
  écris au 18,10;"1      2      3      4      5      6      7      8      9"
  écris au 21,10;"FIABLE - PEU FIABLE";
  choix;"123456789"
  que C8$=ch$
  écris au 21,10;"RAPIDE - LENT    ";
  choix;"123456789"
  que C9$=ch$
```

```
écris au 21,10;"FACILE - COMPLIQUE   ";
choix;"123456789"
que C10$=ch$
écris au 21,10;"UTILE - INUTILE      ";
choix;"123456789"
que C11$=ch$
quest8
fproc
```

Ce genre de questions est fréquent dans les questionnaires et généralement pénible à dépouiller ; nous verrons cela d'ici quelques pages. En ce qui concerne la structure, rien d'extraordinaire, c'est encore **choix** qui s'occupe de saisir les réponses et de contrôler leur validité. Chaque échelle occupe un champ (C8$ à C11$).

```
proc quest8
  éponge
  écris "QUESTION 8"
  écris
  écris "POUVEZ VOUS QUANTIFIER EN POURCENTAGE VOTRE UTILISATION ?"
  écris au 5,10;"UTILISATION PROFESSIONNELLE (%) ";:quant
  que C12$=qt$
  écris au 7,10;"UTILISATION DOMMESTIQUE (%) ";:quant
  que C13$=qt$
  écris au 9,10;"AUTRE UTILISATION (%) ";:quant
  que C14$=qt$
  écris au 21,10;"          QUESTIONNAIRE TERMINE          ";
  boucle;20
  menu
  fproc
```

La question 8 fait appel à la procédure **quant**, qui peut aussi servir dans le cas de pourcentages.

Nous arrivons maintenant à la fin de ce petit questionnaire, qui aurait pu, bien entendu, continuer encore longtemps, mais, vous l'avez vu, la plupart des questions font appel aux procédures de bases que nous avons décrites. Peu importe le libellé de la question, c'est le mode de saisie et l'analyse qui réclame le plus de travail.

Une fois que la question 8 a été remplie, nous retournons, après un message signalant que le questionnaire est terminé, au menu.

Là, vous avez plusieurs solutions, soit enregistrer les informations que vous venez de saisir — c'est-à-dire les **ajouter** à la base de donnée — ou bien les ignorer et recommencer un nouveau questionnaire, et dans ce cas, toutes les informations que vous aurez saisies seront perdues. L'utilisation normale veut que l'on enregistre les informations et que l'on passe au suivant (CHOIX 1).

Vous pouvez aussi, si vous avez terminé une session, analyser les

données saisies ; dans ce cas, deux solutions : entamer une procédure d'analyse (sans enregistrer les informations qui viennent d'être saisies), ou bien les enregistrer et passer à la procédure d'analyse (CHOIX 4) ; l'utilisation normale veut que cela soit ce choix que l'on sélectionne lorsque l'on vient de saisir des informations.

5.3 LE DÉPOUILLEMENT

Quelle que soit l'option choisie, la même procédure est appelée ; il s'agit suivant toute logique de la procédure **analyse**.

```
proc analyse
  éponge
  écris au 7,10;"LES DIFFERENTES METHODES : "
  écris au 10,20;"(E)CHELLES (!----!----!----!)"
  écris au 11,20;"(O)UI / NON"
  écris au 12,20;"(C)HOIX MULTIPLES"
  écris au 13,20;"(S)EGMENTS"
  écris au 14,20;"(G)ENERAL"
  écris au 15,20;"(L)IBRE"
  écris au 16,20;"(M)ENU"
  choix;"EOCSGLM"
  que chx$=ch$
  si chx$="E":analech: fsi
  si chx$="O":analoui: fsi
  si chx$="C":analmul: fsi
  si chx$="S":analpour: fsi
  si chx$="G":analgene: fsi
  si chx$="L":anallib: fsi
  si chx$="M":menu: fsi
  fproc
```

Cette procédure n'est rien d'autre qu'un second menu, dont les choix concernent cette fois les méthodes d'analyse du questionnaire.

Différentes options :

```
proc analech
  mode 0
  que z=0
  éponge
  saisis au 5,10;"COMBIEN Y A T IL DE CHAMPS CONCERNES ? ";nbr
  éponge
  tantque z<nbr
    que z=z+1
    écris au 21,20;"ENTREZ LE NOM DU CHAMP ";z;"  : ";: saisis chp$
    que y=0
```

```
  tantque y<9
    que tot=0
    que y=y+1
    partout
      si val(valvar(chp$))=y: que tot=tot+1: fsi
      fpartout
    écris au z*4,0;chp$
    écris au z*4,10;"1      2      3      4      5      6      7      8      9"
    écris au 1+z*4,10;"!------!------!------!------!------!------!------!------!"
    écris au 2+z*4,2;"NBRE :"
    écris au 2+z*4,(y*7)+3;tot
    écris
    ftantque
  ftantque
suite
fproc
```

Puisqu'il y a des questions en échelles, il est normal qu'il y ait un dépouillement en échelles ! Cette procédure est utilisable dans la majorité des cas, puisqu'elle est entièrement paramétrable (nombre de champs concernés, nom du champ) et ARCHIVE fait le reste. Les résultats viennent s'afficher juste en dessous de chaque « barreau » de l'échelle. Le seul élément qui ne soit pas paramétrable, c'est la longueur de l'échelle et le nombre de barreaux, ce qui est facilement modifiable. Vous pouvez par exemple remplacer la ligne **tantque y<9** par n'importe quelle valeur de votre choix, et bien sûr adapter l'échelle au nombre de barreaux que vous voulez y mettre.

```
proc analoui
  que tot1=0: que tot2=0: que tot3=0: que tot4=0: que base=0
  éponge
  saisis au 5,10;"QUEL CHAMP EST CONCERNE ? ";chp$
  écris au 6,10;"QUELLE BASE ? (";nombenr();")";: saisis base
  si base=0: que base=nombenr(): fsi
  éponge
  partout
    si valvar(chp$)="1"
      que tot1=tot1+1
      fsi
    si valvar(chp$)="2"
      que tot2=tot2+1
      fsi
    si valvar(chp$)="3"
      que tot3=tot3+1
      fsi
    si non souschn("123",valvar(chp$))
      que tot4=tot4+1
      fsi
    fpartout
```

```
que base=base-tot4
éponge
écris au 5,10;"NOMBRE DE PERSONNES AYANT REPONDU 'OUI' : ";tot1;"  ";tot1/base*100;"%"
écris au 7,10;"NOMBRE DE PERSONNES AYANT REPONDU 'NON' : ";tot2;"  ";tot2/base*100;"%"
écris au 9,10;"NOMBRE DE NSP : ";tot3/base*100;"%"
écris au 12,10;"PAS DE REPONSE : ";tot4
suite
fproc
```

Analyse classique du nombre de personnes ayant répondu OUI, NON, NSP. Les résultats étant donnés bruts et en pourcentages, une base est demandée. Si vous ne souhaitez pas de base spécifique, tapez simplement sur <RETURN> et c'est le nombre total d'enregistrements qui servira de base. A noter cependant que cette dernière est corrigée des enregistrements qui à la suite d'une mauvaise manœuvre n'auraient reçu aucune information et fausseraient par conséquent les résultats en pourcentages.

```
proc analmul
  que z=0: que lim=1
  éponge
  saisis au 5,10;"QUEL EST LE CHAMP CONCERNE ? ";chp$
  note saisis au 7,10;"COMBIEN Y A T IL DE CHOIX POSSIBLES ? ";chx
  éponge
  écris au 5,0;"UN INSTANT..."
  tantque z<lim
    que z=z+1
    que tot1=0
    partout
      que r=0
      tantque r<long(valvar(chp$))
        que r=r+1
        si lim<val(valvar(chp$)(r jusq r)): que lim=val(valvar(chp$)(r jusq r)): fsi
        si val(valvar(chp$)(r jusq r))=z
          que tot1=tot1+1
          fsi
        ftantque
      fpartout
    écris tab 10;"NOMBRE DE PERSONNES POUR LE CHOIX NO :";z;"  ";tot1
    ftantque
  suite
  fproc
```

C'est la procédure qui réclame le plus de temps en traitement, mais qui en économise de toute façon par rapport à un dépouillement manuel !

ARCHIVE va scruter sur l'ensemble du fichier chacune des réponses contenues dans le champ spécifié. Remarquez que l'analyse ne va que jusqu'à une limite déterminée automatiquement par ARCHIVE.

Exemple : dans la question 5, il y a sept possibilités de réponses. ARCHIVE commence par rechercher le nombre de personnes ayant choisi 1 comme raison, puis 2, puis 3, etc. Mais si personne n'a répondu au choix 6 et 7, l'analyse s'arrêtera à 5 ; ceci pour gagner du temps. Ce format peut être modifié, si ce n'est pas celui que vous souhaitez ; donnez alors comme valeur à **lim** (1re ligne) le choix maximum (ici 7) et en supprimant la ligne :

si lim<valvar(chp$) (r jusq r):que lim=val(valvar(chp$)(r jusq r)): fsi

```
proc analpour
  mode 0
  que pas=0: que lim=0
  éponge
  saisis au 5,10;"COMBIEN Y A T IL DE CHAMPS CONCERNES ? ";nbr
  saisis au 7,10;"QUEL PAS ? (10 EN 10):";pas
  saisis au 9,10;"SUR COMBIEN DE SEGMENTS ? (10) ";lim
  si lim=0: que lim=10: fsi
  si pas=0: que pas=10: fsi
  que z=0
  éponge
  tantque z<nbr
    que z=z+1
    écris au 21,20;"ENTREZ LE NOM DU CHAMP ";z;"  : ";: saisis chp$
    éponge
    écris au 5,10;"CHAMP : ";chp$: écris
    que x1=0: que x2=0
    tantque x1<lim*pas
      que x1=x1+pas
      que x2=x1-pas
      que tot1=0
      partout
        si val(valvar(chp$))<=x1 et val(valvar(chp$))>x2
          que tot1=tot1+1
          fsi
        fpartout
      écris tab 10;"NBRE DE PERSONNES ENTRE ";x2;" ET ";x1;"  ";tot1
      ftantque
    ftantque
  suite
  fproc
```

Procédure avant tout destinée à analyser les pourcentages, mais qui peut aussi servir à toutes les analyses par segment. Les pourcentages, comme les quantités, ne peuvent s'analyser sur une moyenne (doit-on dire qu'en moyenne les personnes interrogées possèdent 1.2 ordinateurs ?!?). En conséquence, cette procédure,

encore une fois, entièrement paramétrable, peut être utilisée dans ces cas.

La première question concerne le nombre de champs que l'on veut analyser de cette manière, la seconde demande le pas (10 en 10 par défaut), et la troisième le nombre de segments (10 par défaut). Ensuite seulement vous êtes invité à rentrer le nom des champs concernés un à un.

```
proc analgene
  que base=0: que x=0: que lim=1
  éponge
  saisis au 5,10;"QUELLE CHAMP VOULEZ VOUS ANALYSER ? ";chp$
  écris au 7,10;"QUELLE BASE ? (";nombenr();") ";: saisis base
  si base=0: que base=nombenr(): fsi
  éponge : écris au 3,10;"CHAMP : ";chp$: écris
  tantque x<lim
    que x=x+1
    que tot1=0
    partout
      si val(valvar(chp$))=x: que tot1=tot1+1: fsi
      si val(valvar(chp$))>lim: que lim=val(valvar(chp$)): fsi
      fpartout
    écris tab 10;"NOMBRE DE PERSONNES AYANT REPONDU ";x;" : ";tot1;"   ";tot1/base*100;"%"
    ftantque
  suite
  fproc
```

Cinquième choix du menu des analyses, ce format vous permet d'analyser tous les cas qui ne rentrent pas dans les procédures que nous venons de voir. L'analyse se fait en données brutes et en pourcentages sur un champ que vous devez préciser.

```
proc anallib
  éponge
  saisis au 5,10;"QUEL CHAMP ? ";chp$
  saisis au 7,10;"QUELLE CHAINE RECHERCHEZ VOUS ? :";chain$
  écris au 9,10;"UN INSTANT..."
  partout
    si souschn(valvar(chp$),chain$)
      que tot1=tot1+1
      fsi
    fpartout
  éponge
  écris au 5,10;"CHAMP : ";chp$;"         CHAINE : ";chain$
  écris au 7,10;"REPEREE DANS ";tot1;" QUESTIONNAIRES"
  suite
  fproc
```

Cette procédure est extrêmement puissante. Le dépouillement des questions ouvertes demande en général un très gros travail de

regroupement sous des rubriques plus ou moins larges. Ici, il suffit de préciser la chaîne que l'on désire et ARCHIVE se charge de dire combien de fois elle a été mentionnée.

A la fin de chacune de ces analyses, la procédure **suite** est systématiquement appelée ; elle offre la possibilité de revenir au menu ou de continuer l'analyse.

```
proc suite
  écris au 21,10;"          (M)ENU - (A)NALYSE          ";
  que chx$=majus(clavier())
  si chx$="M":menu: fsi
  analyse
  fproc
```

C'est une utilisation un peu inhabituelle des logiciels de bases de données que nous avons décrite ici, mais la puissance d'ARCHIVE — et en particulier de son langage de programmation — en fait un outil extrêmement souple, surtout en ce qui concerne ce genre d'applications facilement exécutable par un outil informatique.

ANNEXE A :

Programme "BOOT"

Le programme ci-dessous permet aux possesseurs de QL de pouvoir charger ARCHIVE, QUILL, ABACUS ou EASEL à partir d'un menu central. Vous pouvez le sauvegarder sur vos cassettes à la place du programme "boot" original. Attention, ce programme ne peut être utilisé qu'avec les versions 2 des logiciels.

```
100 REMark Menu g|n|ral pour le QL "BOOT"
110 CLEAR
120 WINDOW 512,256,0,0:MODE 0:CLS:CSIZE 2,1
130 CLS
140 REPeat boucle
150 AT 1,15:PRINT "MENU"
160 AT 3,10:PRINT "F1 - ABACUS"
170 AT 4,10:PRINT "F2 - EASEL"
180 AT 5,10:PRINT "F3 - ARCHIVE"
190 AT 6,10:PRINT "F4 - QUILL"
200 AT 7,10:PRINT "F5 - BASIC"
210 AT 9,10:PRINT "VOTRE CHOIX ?";
220 chx=CODE(INKEY$(-1))
230 SELect ON chx
240    =232
250     charge "ABACUS","TABLEUR"
260    =236
270     charge "EASEL","GRAPHIQUES"
280    =240
290     charge "ARCHIVE","SGDB"
300    =244
```

```
310     charge "QUILL","TEXTES"
320    =248
330     NEW
340    =REMAINDER
350     BEEP 1000,10
360 END SELect
370 END REPeat boucle
380 DEFine PROCedure charge (nom$,application$)
390 CLS
400 AT 2,2:PRINT "METTEZ LA CASSETTE CONTENANT"!nom$!
410 AT 3,2:PRINT "DANS LE LECTEUR DE GAUCHE"
420 AT 5,2:PRINT "ET TAPEZ SUR UNE TOUCHE POUR COMMENCER"
430 PAUSE
440 CLS
450 AT 2,8:PRINT "CHARGEMENT DE :"!nom$
460 AT 4,15:PRINT application$
470 AT 7,12:PRINT "VERSION 2.00"
480 AT 9,8:PRINT "Copyright PSION Ltd."
490 SELect ON chx
500   =232
510    CLOSE#1:CLOSE#2:WINDOW#0,400,20,35,215
520 EXEC_W mdv1_abacus
530   =236
540    CLOSE#1:CLOSE#2:WINDOW#0,400,20,35,215
550    EXEC_W mdv1_easel
560   =240
570    CLOSE#1:CLOSE#2:WINDOW#0,400,20,35,215
580    EXEC_W mdv1_archive
590   =244
600    CLOSE#1:CLOSE#2:WINDOW#0,400,20,35,215
610    EXEC_W mdv1_quill
620 END SELect
630 OPEN#1,scr:OPEN#2,scr:INK#1,7:INK#2,7
640 RUN
```

ANNEXE B :

Programme de publi-postage pour le QL

Ce programme vous permet d'utiliser conjointement un document écrit avec QUILL et un fichier conçu avec ARCHIVE. Vous pouvez remplacer n'importe quelle expression contenue dans QUILL par un des éléments du fichier ARCHIVE.

En premier lieu le programme vous demande de lui spécifier un certain nombre de renseignements afin que le déroulement des opérations se passe correctement :

* Le lecteur où se trouve le fichier ARCHIVE

Vous devez répondre 1 ou 2 puis valider votre choix

* Le nom du fichier

Vous devez rentrer le nom du fichier complètement, c'est-à-dire avec son extension. Attention, si vous n'avez pas spécifié d'extension, celle-ci sera **_lis**, étant donné que les fichiers à exploiter sont « fabriqués » comme s'ils étaient prêts pour l'impression.

* Le nombre de champs

Ce nombre doit correspondre au nombre de champs **par enregistrement**.

Aussitôt la structure du fichier ARCHIVE s'affiche à l'écran ainsi que le contenu du premier enregistrement.

Vous devez alors taper sur une touche pour passer au document QUILL.

Les deux premières informations demandées sont les mêmes

que précédemment, à savoir le lecteur et le nom du document QUILL (n'oubliez pas l'extension), puis c'est le nombre de copies que vous désirez que vous êtes invités à rentrer. Ce nombre doit être égal au nombre d'enregistrements contenus dans le fichier ARCHIVE.

Une fois ces opérations effectuées, vous devez indiquer les expressions du document QUILL que vous voulez remplacer par les informations du fichier ARCHIVE. Vous pouvez remplacer un maximum de 10 expressions. Pour que les justifications soient respectées, nous vous conseillons de prévoir lors de l'écriture du document QUILL un nombre de lettres supérieur au contenu du fichier ARCHIVE.

Par exemple, s'il s'agit d'un nom :

Monsieur &nnnnnnnnnnnnnnnnnn

où &nnnnnnnnnnnnnnnnn représente l'expression à remplacer par le champ **nom** du fichier ARCHIVE. Attention, lorsque le programme vous demandera quel mot vous désirez substituer, vous devrez indiquer &nnnnnnnnnnnnnnnnn dans son intégralité en ayant soin de n'omettre aucune lettre, sinon l'expression ne serait pas entièrement substituée. N'oubliez pas de prévoir un nombre de caractères supérieur aux informations contenues dans le champ du document ARCHIVE, autrement une erreur serait générée.

Vous devrez ensuite indiquer la position du champ dont les informations doivent remplacer cette expression (la position est indiquée à la gauche de chaque champ du fichier ARCHIVE). Ces deux informations apparaissent alors à l'écran, et vous pouvez continuer à indiquer d'autres expressions à substituer ou bien arrêter en tapant "STOP" après la question "Quel mot du document QUILL voulez-vous remplacer". Une fois ceci fait, vous devez taper sur une touche pour lancer le processus d'impression.

Nous avons réglé la longueur de page à 66 lignes par page, mais vous pouvez modifier ceci en modifiant la ligne 1500.

Voici un exemple de document QUILL :

FDS EDIMICRO
121-127 Avenue d'Italie
75013 PARIS

Paris, le 25 octobre 1985

&nnnnnnnnnnnnnnnnnnn
&aaaaaaaaaaaaaaaaaaaa
&ccccc &vvvvvvvvvvv

&pppppppppppp,

Nous vous rappelons que votre dossier &rrrrrrrrrrrrrrr est toujours en cours de traitement et que nous vous contacterons rapidement. Nous vous demandons toutefois de nous confirmer que votre numéro de téléphone est bien le &tttttttttttt.

Dans l'attente d'une réponse de votre part, nous vous prions de croire, &pppppppppppp en nos meilleurs sentiments.

FDS EDIMICRO

Cette lettre devra correspondre à un fichier ARCHIVE contenant les informations suivantes :

ARCHIVE	QUILL
(2) nom$	&nnnnnnnnnnnnnnnnnnn
(3) adresse$	&aaaaaaaaaaaaaaaaaaaa
(4) code_post$	&ccccc
(5) ville$	&vvvvvvvvvvvv
(6) patronyme$	&pppppppppppp
(7) ref$	&rrrrrrrrrrrrrrr
(8) tel$	&tttttttttttt

Les deux fichiers doivent être « imprimés » sur un fichier. C'est-à-dire qu'ils porteront l'extension **.lis** (sauf ordre contraire de votre part).

Ceci s'effectue avec la commande **I**mprime de QUILL et **via** d'ARCHIVE.

Exemple pour le fichier ARCHIVE :

```
proc publi
  saisis "NOM DU FICHIER";nom$
  via nom$
  imprime
  début
  tantque eof()<1
    imprime nom$
    imprime adresse$
    imprime code_post$
    imprime ville$
    imprime patronyme$
    imprime ref$
    imprime tel$
    suiv
  ftantque
  normal
fproc
```

Vous pouvez, bien sûr, selon la même manière, créer des fichiers sélectifs en **isolant** certains enregistrements qui correspondent à un critère de votre choix.

```
100 init
110 DEFine PROCedure init
120     DIM fich$(500,50),docu$(200,100),mot$(
10,30),pos(10)
130     MODE 0:PAPER 4:CLS:CLS#0
140 END DEFine init
150 document
160 fichier
170   PRINT#0,"APPUYEZ SUR UNE TOUCHE POUR CO
MMENCER"
180   PAUSE
190   CLS#0
200   BEEP 1000,10
210 menu_archive
220 menu_quill
230 DEFine PROCedure document
240  INK 7
250  BLOCK 150,150,30,30,0
260  BLOCK 150,150,24,24,2
270  LINE 15,83 TO 25,83
280  LINE 15,81 TO 25,81
290  LINE 15,79 TO 25,79
300  LINE 50,70 TO 60,70
310  LINE 50,68 TO 60,68
```

```
320  LINE 50,66 TO 60,66
330  LINE 40,60 TO 60,60
340  LINE 20,50 TO 30,50
350  LINE 20,40 TO 60,40
360  LINE 15,38 TO 60,38
370  LINE 15,36 TO 60,36
380  LINE 15,34 TO 60,34
390  LINE 15,32 TO 60,32
400  LINE 15,30 TO 60,30
410  LINE 40,25 TO 55,25
420 END DEFine document
430 DEFine PROCedure fichier
440  BLOCK 150,150,272,30,0
450  BLOCK 150,150,266,24,2
460   FOR i=1 TO 140 STEP 20
470    BLOCK 50,8,276,30+i,0
480    BLOCK 50,8,272,28+i,7
490   END FOR i
500 END DEFine fichier
510 DEFine PROCedure menu_quill
520  CLS
530  FOR i=10 TO 150 STEP 5
540    BLOCK i,i,i/5,i/5,2
550    BLOCK i,i,i/5,i/5,4
560  END FOR i
570  document
580   INK 0:STRIP 7
590   BLOCK 250,150,192,30,0
600   BLOCK 250,150,186,25,7
610    AT 3,40:PRINT "DOCUMENT QUILL"
620    AT 5,33:INPUT "LECTEUR OU SE TROUVE LE
 FICHIER :"!lect_b$
630    AT 7,33:INPUT "QUEL EST LE NOM DU FICH
IER :"!nom_b$
640    AT 9,33:INPUT "COMBIEN DE COPIES DESIR
EZ VOUS "!nbr_copies$
650 document_quill
660 remplacement
670 END DEFine menu_quill
680 DEFine PROCedure document_quill
690  OPEN_IN#4,"mdv"&lect_b$&"_"&nom_b$
700  lim_1=0
710    REPeat boucle_quill
720     lim_1=lim_1+1
730      IF lim_1>66 THEN EXIT boucle_quill
740     INPUT#4,docu$(lim_1)
760 END REPeat boucle_quill
```

```
770 END DEFine document_quill
780 DEFine PROCedure menu_archive
790 CLS:INK 0:STRIP 7:i=0
800 fichier
810     i=0
820     BLOCK 250,150,8,30,0
830     BLOCK 250,150,2,24,7
840      AT 3,13:PRINT "FICHIER ARCHIVE"
850      AT 5,3:INPUT "LECTEUR OU SE TROUVE LE
 FICHIER :"!lect_a$
860      AT 7,3:INPUT "NOM DU FICHIER :"!nom_a
$
870      OPEN#3,"mdv"&lect_a$&"_"&nom_a$
880      AT 9,3:INPUT "COMBIEN Y-A-T-IL DE CHA
MPS :"!nbr_champs$
890    REPeat boucle_a
900     i=i+1
910     IF EOF(#3) THEN EXIT boucle_a
920       INPUT#3,fich$(i)
930    END REPeat boucle_a
940    BLOCK 260,160,2,24,4
950   structure
960   PRINT#0,"APPUYEZ SUR UNE TOUCHE POUR CON
TINUER"
970   PAUSE
980   CLS#0
990 END DEFine menu_archive
1000 DEFine PROCedure structure
1010    BLOCK 150,150,270,30,0
1020    BLOCK 150,150,264,24,7
1030    BLOCK 60,10,220,25,0
1040    BLOCK 60,10,214,21,7
1050    BLOCK 60,10,320,21,0
1060    BLOCK 60,10,314,15,7
1070    CURSOR 316,15:PRINT "CONTENU"
1080    CURSOR 216,21:INK 0:PRINT "POSITION"
1090      FOR i=2 TO nbr_champs$+1
1100       CURSOR 280,10+(i*20):STRIP 0:INK 0:
PRINT i;"  ";fich$(i)
1110       CURSOR 274,6+(i*20):STRIP 2:INK 7:P
RINT i;"  ";fich$(i)
1120      END FOR i
1130 END DEFine structure
1150 DEFine PROCedure remplacement
1160  CLS
1170 structure
1180    lim_2=0
```

```
1190    REPeat boucle_remp
1200     lim_2=lim_2+1
1210      PRINT#0, "QUEL MOT DU DOCUMENT QUIL
L VOULEZ VOUS REMPLACER ('STOP' POUR ARRETER)
"
1220      INPUT#0,mot$(lim_2)
1230       IF mot$(lim_2)="stop" OR mot$(lim_
2)="STOP" THEN EXIT boucle_remp
1240        FOR j=1 TO lim_1
1250         IF mot$(lim_2) INSTR docu$(j)
1260            AT 3+i,5:STRIP 7:INK 0:PRINT
mot$(lim_2);
1270            INPUT#0,"A REMPLACER PAR LE C
ONTENU DU CHAMP EN POSITION ";pos(lim_2)
1280            CURSOR 280,10+(pos(lim_2)*20)
:STRIP 0:INK 0:PRINT pos(lim_2);"  ";fich$(po
s(lim_2))
1290            CURSOR 274,6+(pos(lim_2)*20):
STRIP 4:INK 0:PRINT pos(lim_2);"  ";fich$(pos
(lim_2))
1300            BEEP 1000,20:PAUSE 50
1310            CURSOR 274,6+(pos(lim_2)*20):
STRIP 2:INK 7:PRINT pos(lim_2);"  ";fich$(pos
(lim_2))
1320         END IF
1330        END FOR j
1340  AT 5+i,15:STRIP 0:INK 7:PRINT fich$(pos
(lim_2))
1350 END REPeat boucle_remp
1360 execution
1370 END DEFine remplacement
1380 DEFine PROCedure execution
1390  CLS#0:PRINT#0,"ASSUREZ VOUS QUE L'IMPRI
MANTE EST EN ORDRE ET TAPEZ SUR UNE TOUCHE"
1400  PAUSE
1410   FOR a=0 TO nbr_copies$
1420    FOR b=1 TO lim_2
1430     FOR c=1 TO lim_1
1440       IF mot$(b) INSTR docu$(c) THEN
1450         docu$(c)=docu$(c)(1 TO mot$(b) I
NSTR docu$(c)-1)&fich$(pos(b)+(a*nbr_champs$)
)&FILL$(" ",LEN(mot$(b))-LEN(fich$(pos(b)+(a*
nbr_champs$))))&docu$(c)(mot$(b) INSTR docu$(
c)+LEN(mot$(b)) TO LEN (docu$(c)))
```

```
1460        END IF
1470      END FOR c
1480    END FOR b
1490                          OPEN#10,ser
1500                            FOR l=1 TO 66
1510                             PRINT#10,docu$(l)
1520                            END FOR l
1530 document_quill
1540   END FOR a
1550 PRINT docu$
1560 CLOSE#3
1570 CLOSE#4
1580 CLOSE#10
1590 END DEFine execution
```

ANNEXE C :

Quelques définitions de vocabulaire

AFFECTER
Se dit d'une variable que l'on initialise avec une valeur quelconque. En ARCHIVE, toutes les variables doivent être initialisées.

ALPHANUMÉRIQUE
Se dit d'une variable ou d'un champ destiné à recevoir du texte.

CATALOGUE
Lister le contenu de la mémoire de masse (cassettes ou disquettes). La commande permettant de faire ceci est : **cat** "nom_du_lecteur".

CHAMP
ou ZONE. Les champs sont les différentes rubriques qui composent un enregistrement.

COMMANDE
Une commande est un mot-clé reconnu par ARCHIVE et tapé dans le mode direct.

DEFAUT
On parle de valeur prise par « défaut ». C'est la valeur prise

automatiquement par ARCHIVE si rien ne lui est précisé. Par exemple « le lecteur par défaut », « l'extension par défaut ».

EXTENSION

Une extension est un suffixe ajouté automatiquement par ARCHIVE. Nous donnons la liste des extensions en annexe. Exemple SCN, qui indique que le fichier correspond à un masque d'écran.

FICHES/ENREGISTREMENTS

Nous avons souvent confondu les deux termes. En fait enregistrement est plus correct. Il s'agit de l'information contenue par un ensemble de champs (ou de zones).

INDEX

Lorsque l'on trie un fichier, celui-ci se voit donné un numéro d'index. Il est alors possible de rappeler un tri particulier en fonction de l'index. Ceci n'est disponible que sur l'IBM PC.

INSTRUCTION

Une instruction est un mot-clé, reconnu par ARCHIVE et mis dans un programme.

MEMOIRE DE MASSE

Mémoire qui enregistre les informations de façon permanente par opposition à la mémoire vive (RAM), qui disparaît lorsque l'on éteint l'appareil. Les disquettes et cassettes sont des mémoires de masse.

NOM LOGIQUE

Tous les fichiers ouverts sous ARCHIVE possèdent un « nom physique » et un « nom logique ». Ce dernier est obligatoire lorsque plusieurs fichiers sont ouverts en même temps.

NOM PHYSIQUE

Le nom physique est le nom donné aux fichiers lorsque vous les sauvegardez. C'est celui-ci que vous devez spécifier lorsque vous réouvrez un fichier.

NUMERIQUE

Se dit d'une variable ou d'un champ destiné à recevoir des nombres.

ORDRE HISTORIQUE

Ordre dans lequel les informations d'un fichier ont été rentrées la première fois.

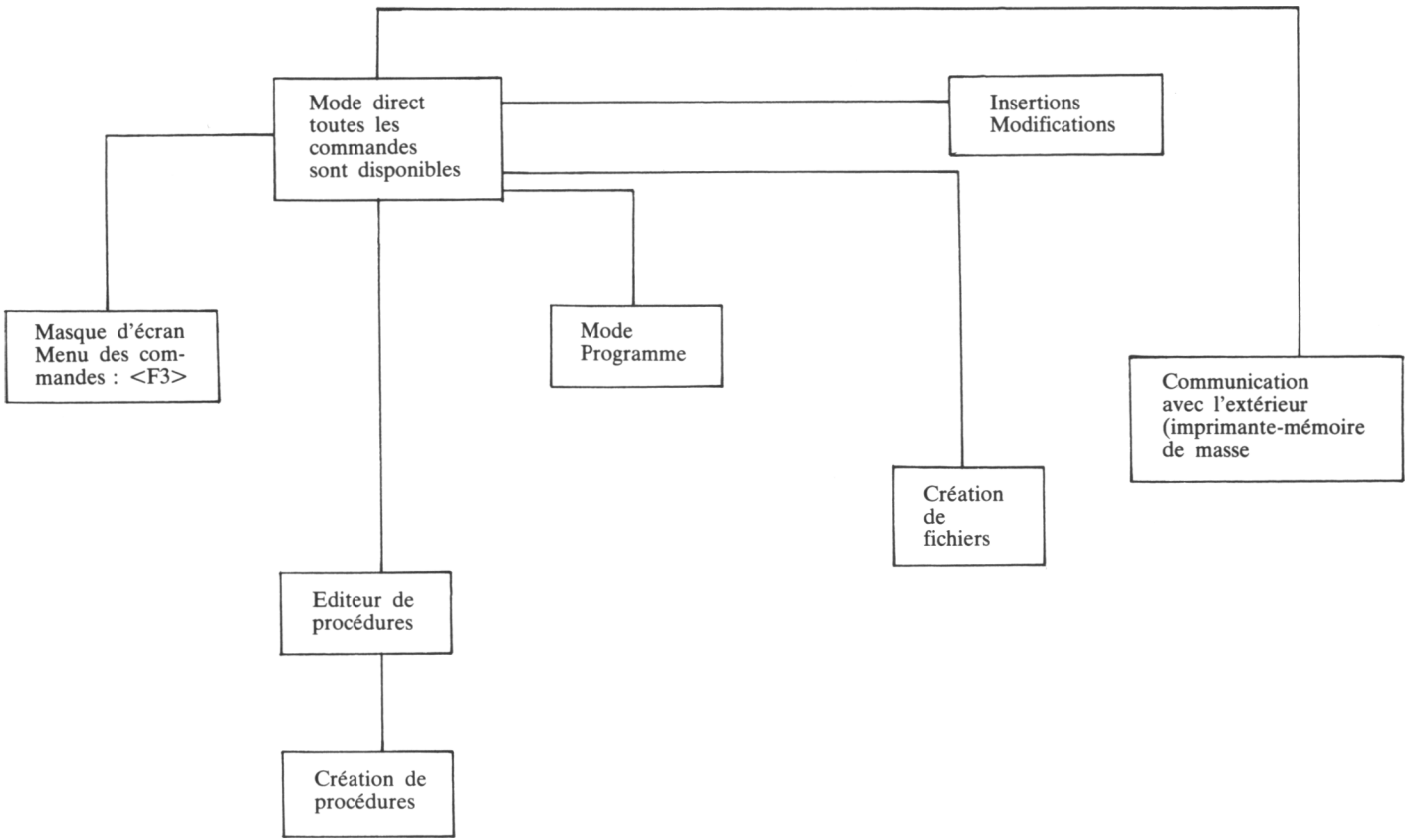
PROMPT

Signe apparaissant à l'écran et marquant l'attente d'une commande de la part de l'utilisateur. En mode direct le prompt (petit rectangle) est toujours présent dans ARCHIVE.

LISTE DES EXTENSIONS DES FICHIERS CRÉÉS PAR ARCHIVE

Programmes	PRG
Programmes objets	PRO
Masque d'écran	SCN
Bases de données	DBF
Fichiers imprimables	LIS
Exportation	EXP

ANNEXE D : Les différents modes d'ARCHIVE



LA BIBLIOTHÈQUE EDIMICRO

- ***Collection « Ordinateurs professionnels »***

APPLE

de Merly ..... **GUIDE DE L'APPLE (3 tomes)**
Tome 1 : l'Apple standard
Tome 2 : les extensions
Tome 3 : les applications

MACINTOSH

Gaucherand ..... **MACINTOSH : outils, progiciels, applications**
Longevialle ..... **Programmez en PASCAL SUR MACINTOSH**

SINCLAIR QL

Tenin, Van Thong ..... **Guide pratique du Sinclair QL**
K et S Brain ..... **Intelligence artificielle sur QL**
Tenin ..... **ARCHIVE sur Sinclair QL**
Devisscher ..... **ABACUS, EASEL et QUILL sur Sinclair QL**

- ***Collection « Progiciels »***

animée par MM. Ph. Bonnet et M.T. Dinh

APPLE

Bonnet, Dinh ..... **Multiplan sur Apple**

IBM PC

Bonnet, Dinh ..... **Lotus 1-2-3 à votre portée**
Bonnet, Dinh ..... **Multiplan sur IBM PC : exercices de gestion**
Commandeur ..... **Framework sur IBM PC**

MACINTOSH

Bouilloux ..... **Multiplan et Chart sur Macintosh**
Hornn ..... **JAZZ sur Macintosh**
Bouilloux ..... **EXCEL sur Macintosh**
Bouilloux ..... **CX MacBase 500**
Joutel, Tenin ..... **Les Bases de données sur Macintosh**

GENERAL

Bonnet, Dinh ..... **Mémento Multiplan**
Hoenig ..... **Tout sur FRAMEWORK**

- ***Collection « Langages »***

Gaucherand, Lamoitier ..... **Fichiers en Basic par l'exemple**
Perbost, Berthet ..... **Introduction à MSX**

- ***Collection « Ouvrages de base »***

Lamoitier ..... **Le Traducteur Micro**
Darnis, Van Thong ..... **Graphisme et CAO**
Avon, Rolet ..... **Courbes de maths en Basic**

- ***Collection « Ordinateurs familiaux »***

VG 5000
Amsler, Bardon ..... **Guide du VG 5000 Philips**
Amsler, Villemaud ..... **Jeux sur VG 5000 Philips**

MO5
Bieber, Perbost, Renucci ..... **Tout sur le MO5**
Perbost, Renucci ..... **Jeux sur MO5**

AMSTRAD
Penfold ..... **L'Amstrad avec plaisir**
Waugh ..... **Musique sur Amstrad**

ATMOS/ORIC 1
Kosniowsky ..... **Nouveaux Jeux sur ATMOS**
Chane-Hune, Darbois ..... **Guide de l'Oric**
Bayvejiel ..... **Jeux Graphiques sur ATMOS**
Viguier ..... **Premiers Pas en Programmation sur Oric**

SPECTRUM
Bridge, Carnell ..... **Aventures sur Spectrum**
Hurley ..... **Jeux Graphiques sur Spectrum**

COMMODORE 64
Fleurier, Meiller ..... **Jeux sur Commodore 64 : Jeux d'adresse et de hasard**
Fleurier, Meiller ..... **Jeux sur Commodore 64 : Jeux d'action et de réflexion**

ELECTRON/BBC
Bennani, Chaieb ..... **Graphisme et sons sur Electron et BBC**

PHILIPS C7420 VIDEOPAC +
Bardon, de Merly ..... **Jeux sur Philips C7420 Vidéopac +**

TO7
— ouvrages
Bieber, Perbost, Renucci ..... **Guide du TO7**
Perbost, Renucci ..... **Jeux sur TO7**

— logiciels sur cassette
Perbost, Renucci ..... **4 jeux avec manette sur TO7**
Perbost, Renucci ..... **6 jeux : action et réflexion sur TO7**

GENERAL

de Merly ..... **Ordinateur familial : Que choisir ?**
Djama ..... **Je construis mon premier robot**
Ly ..... **Applications familiales en Basic MSX**
Perbost, Berthet ..... **Introduction à MSX**

Nombreux autres titres à paraître. Catalogue sur simple demande.

EDIMICRO 121-127, avenue d'Italie, 75013 Paris

Achevé d'imprimer en novembre 1985
sur les presses de l'imprimerie Laballery et Cie — 58500 Clamecy
Dépôt légal : novembre 1985 N° d'imprimeur : 509095

EXTRAITS DE PRESSE

COURBES DE MATHS EN BASIC

« Constitue une excellente introduction au graphisme..., pour réaliser de superbes courbes mathématiques. »

Théophile.

MULTIPLAN ET CHART SUR MACINTOSH

« Présente l'incomparable avantage de lier Microsoft Chart et Microsoft Multiplan... »

Bernard Vergnes, Directeur de Microsoft pour l'Europe.

MULTIPLAN SUR IBM PC

« Ce livre rendra de nombreux services. »

OPC.

« Des modèles de qualité professionnelle. »

L'Ordinateur Personnel.

FICHIERS EN BASIC PAR L'EXEMPLE

« Cet ouvrage est recommandé aux personnes amenées à entreprendre la programmation des fichiers du Basic, pour lesquels il constitue une étude pratique, sérieuse et en profondeur. »

Ordi Magazine.

GUIDE DE L'APPLE

« Un des meilleurs sur la place. »

Le Figaro.

« Le livre que nous attendions : complet, clair et pratique. »

Apple France.

TOUT SUR LE MO5

« Un livre consistant, original et rassurant dans sa démarche... »

Science et Vie Micro.

JEUX SUR VG 5000 PHILIPS

« Ouvrage bien écrit, structuré et clair... »

Microfer.

GUIDE PRATIQUE DU SINCLAIR QL

« Une somme impressionnante d'informations, des programmes nombreux et intéressants... Un bon outil d'initiation à la pratique de notre Sinclair QL. »

Direco International-Sinclair.

« Cet ouvrage comblera tous les possesseurs de l'un des meilleurs ordinateurs du marché. »

Science et Vie Micro.

GUIDE DU TO7

« Ce guide a le privilège d'être présenté de façon claire, dans un style parfaitement accessible à tous. »

Micro 7.

« Un manuel de référence absolue. »

Le Figaro.

JEUX SUR TO7

« De nombreux conseils et "trucs" pour une programmation rapide et efficace. »

Décision Informatique.

« Un excellent livre d'approche, plein de renseignements utiles. »

Micro 7.

JEUX SUR COMMODORE 64

« Cet ouvrage permet aux débutants de s'initier au langage Basic en s'amusant... Un livre d'un bon niveau. »

TILT.

PREMIERS PAS EN PROGRAMMATION

« Une approche originale dans la littérature consacrée aux micros. »

Décision Informatique.

Nombreux autres titres à paraître. Catalogue sur simple demande.

EDIMICRO 121-127, avenue d'Italie, 75013 Paris

ARCHIVE

Progiciel de Bases de Données

E. TENIN

ARCHIVE, progiciel de bases de données extrêmement puissant, est un outil exceptionnel pour gérer vos fichiers, ainsi que toutes les applications qui en découlent.

Ce livre a pour but :

- de vous aider à maîtriser rapidement ARCHIVE par l'exemple, même si vous n'avez jamais touché un ordinateur de votre vie,
- de vous fournir des applications complètes, que vous pourrez réutiliser telles quelles, ou modifier à votre gré : comptabilité, publi-postage, dépouillement d'enquête...
- de vous permettre d'aller plus loin, en apprenant à « programmer » vous-même ARCHIVE.

ARCHIVE est fourni avec le Sinclair QL.
Sous le nom de XCHANGE, il a été adapté pour l'IBM PC. Vous trouverez ici les caractéristiques de cette version ainsi que les programmes pour IBM PC.

Dans la même série :

- GUIDE PRATIQUE DU SINCLAIR QL — 196 pages
- INTELLIGENCE ARTIFICIELLE SUR SINCLAIR QL — 230 pages
- ABACUS, EASEL ET QUILL — 180 pages

ISBN : 2-904457-39-9 135 F.

Edimicro
121-127, Avenue d'Italie 75013 Paris

ARCHIVE

Eric Tenin